Computational Linguistics in the Netherlands 2010

Selected papers from the twentieth CLIN meeting

Published by
LOT
Janskerkhof 13
3512 BL Utrecht
The Netherlands

phone: +31 30 253 6006
fax: +31 30 253 6406
e-mail: lot@let.uu.nl
http://www.lotschool.nl

Cover illustration: Academiegebouw, Utrecht

ISBN: 978-94-6093-047-8
NUR 616

Computational Linguistics in the Netherlands 2010

Selected papers from
the twentieth CLIN meeting
(CLIN 2010)

Edited by:

Eline Westerhout
Thomas Markus
Paola Monachesi

UiL-OTS
Utrecht University

LOT
Utrecht
2010

Preface

The twentieth meeting of CLIN (Computational Linguistics in the Netherlands) took place in Utrecht, The Netherlands, on Friday 5 February 2010. There were 73 abstracts which were accepted for presentation at the conference (55 oral and 18 poster presentations). CLIN 20 attracted an audience of 141 participants. Most of the attendees came from The Netherlands and Belgium, with lower numbers of people from other European countries (5) and the rest of the world (3).

This year we had a special occasion to celebrate: the first CLIN meeting was organized by UiL-OTS 20 years ago and it is becoming a tradition to host it in Utrecht every 10 years.

The Nederlandse Taalunie (Dutch Language Union, DLU) celebrates its 30th anniversary this year. Linde van den Bosch presented on behalf of the DLU a speech entitled '30 years of the Dutch Language Union', which is included in these proceedings.

We were happy to have Bernardo Huberman from the Social Computing Lab at Hewlett-Packard Laboratories as our invited speaker. An abstract of his talk 'Social attention in the age of the web' can be found in these proceedings.

The authors presenting at CLIN have been invited to submit a paper for the proceedings. After a reviewing process, eleven of these papers have been accepted for inclusion in the proceedings.

We would like to thank all people contributing to a successful CLIN meeting: the sponsors, the co-organizers, the organizers of the previous meeting in Groningen, the speakers, the invited speaker, the authors of all submitted papers, the reviewers and everyone that attended CLIN 20.

See you all in Ghent in 2011!

Utrecht, 14 July 2010

Eline Westerhout
Thomas Markus
Paola Monachesi

Organizing Committee

CLIN 20 was organized by:

Eline Westerhout
Thomas Markus
Paola Monachesi

With support from:

Joost Bastings
Coert van Gemeren
Ronald de Haan
Emiel van Miltenburg
Bas Mulders

Sponsors

We would like to thank the following organizations for generously supporting CLIN 20:

Gold sponsors

Silver sponsors

Bronze sponsors

Reviewers

We are grateful to the following reviewers who helped us in the paper selection process for these proceedings:

Toine Bogers
Wauter Bosma
Gosse Bouma
Crit Cremers
Marieke van Erp
Frank van Eynde
Hans van Halteren
Véronique Hoste
Maxime Khalilov
Isa Maks
Thomas Markus
Erwin Marsi
Roser Morante
Gert Jan van Noord
Jan Odijk
Barbara Plank
Martin Reynaert
Ineke Schuurman
Khaled Shaalan
Jörg Tiedemann
Erik Tjong Kim Sang
Shuly Wintner

Contents

Invited talk: Social attention in the age of the web

Bernardo Huberman

Social Computing Lab, Hewlett-Packard Laboratories

Abstract

The past decade has witnessed a momentous transformation in the way people interact, create and exchange information. Content is now coactively produced, shared, classified, and rated on the Web by millions of people, while attention has become an ephemeral and valuable resource that everyone seeks to acquire. This talk will describe our research on the dynamics of attention, how it plays a crucial rule in the generation of content, and the roles that popularity and novelty play in eliciting attention in the Web.

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

30 years of the Dutch Language Union

Linde van den Bosch

Nederlandse Taalunie

It is a great honour for me to address an audience of such renowned computational linguists. I would like to thank the organising committee of CLIN 2010 for this opportunity. The CLIN conference is held every year at a university in either the Netherlands or Belgium. It provides researchers in the field of computational linguistics with a unique opportunity to present and discuss their research. Belgium and the Netherlands are neighbours with much more in common than just a border. What is particularly important in relation to this conference is that the two countries share a common language: Dutch.

It is this bond which, in 1980, led to the two countries joining forces to set up the Dutch Language Union, with the aim of implementing common language policy for Dutch. In 2004, the two founding members were joined by Surinam as an associate member of the Dutch Language Union (DLU).

DLU initiatives cover all aspects of language policy. Each one is aimed at creating the right conditions to make it easier for Dutch speakers to use their language in as many different situations as possible, including when they are abroad.

It is ultimately not the governments of the Netherlands, Flanders and Surinam, who are the DLU's most important 'clients', but the people who use Dutch to communicate.

The activities of the DLU range from promoting the teaching of Dutch, both within the Dutch speaking areas and beyond, and stimulating language-related cultural and literary cooperation, to compiling dictionaries and grammars. Over the last decade the DLU has taken a serious interest in supporting the development of digital language resources and human language technologies (HLT) for Dutch.

As Dutch is a so-called medium-sized language and companies are not always willing or able to invest in developing HLT for a language with a relatively small market, government support was needed. On the other hand, the development of HLT is considered essential, if a language is to survive in the information society.

It was against this background that the DLU set up a number of initiatives aimed at strengthening the position of Dutch in human language technologies. The most important being:

- The STEVIN programme: a subsidised programme aimed at establishing a complete digital language infrastructure for Dutch, and promoting strategic research in language and speech technology
- The HLT Agency: a central repository for digital language resources based

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

within an important linguistic organisation in our language area: the Institute for Dutch Lexicology (INL).

I am delighted that a number of STEVIN results are being presented at this year's CLIN conference.

As the Dutch Language Union is a relatively small organisation, these activities have been set up in close co-operation with the relevant ministries and organisations in Belgium and the Netherlands. This is a key characteristic of the way we work: the DLU does not operate in isolation, but in close cooperation with other professional organisations and associations both within and outside of the Dutch language area. It is thanks to such cooperation that we are today able to look back on the past 30 years, and conclude that a great deal has been achieved, since the DLU was founded in 1980:

- In total there are more than 400,000 learners of Dutch as a foreign language across the world, and Dutch is taught at 200 universities in 40 different countries
- Books by Dutch and Flemish authors are translated into 100 languages
- Close cooperation links have been established with Surinam, the Dutch Antilles, Aruba, South Africa and Indonesia
- Advice on a range a Dutch language and linguistic issues is freely available to the public. In 2009, 5.5 million items were consulted, and 6,300 new questions were submitted and answered
- The DLU website, "Taalunieversum", receives over 17 million visitors a year
- Numerous digital Dutch language resources have been made available to researchers and to the general public. These resources are now being managed and maintained for future use.

2010 marks the 30th anniversary of the DLU, and also the 20th anniversary of the CLIN conference. I would like to congratulate the CLIN community for having reached this milestone. We should acknowledge the valuable contribution CLIN has made towards creating an excellent environment in which Dutch and Belgian computational linguists can exchange and cultivate ideas. It is in part thanks to CLIN that computational linguists, like yourselves, have come to enjoy such international renown.

I would like to conclude with one simple wish: in 10 years' time, when the DLU celebrates its 40th anniversary and CLIN its 30th, my speech will not need to be translated beforehand, but will be able to be produced real-time, using machine translation.

I hope you all enjoy an interesting and fruitful conference.

Improving Successor Variety for Morphological Segmentation

Çağrı Çöltekin

University of Groningen

Abstract

Successor variety is a commonly used measure for segmentation in language processing. It is based on a simple idea that a large variety of letters (or phonemes) following an initial word (or utterance) segment indicates a possible boundary. It dates back to Harris (1955), and several methods based on successor variety have been used in the literature, particularly for the purpose of segmenting words into morphemes. However, there have not been many studies analyzing the measure itself. Even though the idea is simple and effective, the current use in the literature does not utilize the measure to its full extent due to a number of problems with the successor variety scores. This paper intends to address these problems by introducing a normalization method, and demonstrates—using segmentation experiments on two typologically different languages— the effectiveness of this improvement on the morphological segmentation task.

1 Introduction

Segmentation is a prominent problem in language processing. Spoken language input does not contain reliable word boundary markers like white spaces in most writing systems. Even the writing systems that utilize word boundary markers do not mark all linguistically relevant boundaries, such as morpheme boundaries. Humans, as well as computers, need to segment the continuous input into linguistic units such as words and morphemes to be able to interpret the input. This task becomes more important, and more difficult, where the system in question tries to learn these units. In the segmentation task, the competent language users (adult humans or computational systems with linguistic knowledge built in) are aided by rich linguistic information. Although the segmentation task is still difficult, existing linguistic generalizations, e.g., a comprehensive lexicon, are useful for segmentation. However, infants acquiring language or the data driven computational systems do not have that luxury. To be able to acquire a lexicon, learners have to first deal with the segmentation task without a lexicon. This paper analyzes a commonly used measure for segmentation, *successor variety* (SV), first proposed by Harris (1955). We introduce the SV in the context of segmenting written words to morphemes and suggest a simple but effective improvement. To demonstrate the efficiency of our proposed solution, we present a number of experiments on two languages with differing typology.

The successor variety is not a clearly defined algorithm. Several different segmentation algorithms based on the SV have been used with varying success in the literature. Except Hafer and Weiss (1974) and Bordag (2005) there have not been

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

many attempts to analyze and improve the method. This paper first presents an in-depth analysis of the method, then proposes a simple and effective improvement to get more out of it.

Harris (1955) initially proposed the successor variety for segmenting transcribed spoken language utterances into morphemes. The idea is simple: morpheme boundaries are suggested after the utterance segments that may be followed by a large variety of phonemes. In more recent work, however, the measure found its use particularly in segmenting words into morphemes (Hafer and Weiss 1974, Déjean 1998, Al-Shalabi et al. 2005, Bordag 2005, Goldsmith 2006, Bordag 2007, Demberg 2007, Stein and Potthast 2008). Since these studies focused on written text, the measure is frequently referred to as *letter successor variety* (LSV). However, it is equally applicable to other basic (linguistic) units such as phonemes.

The SV based methods are closely related to a number of other approaches based on *entropy*, *(un)predictability*, *surprisal* and *mutual information*. The basic rationale behind these methods is that in a continuous stream formed by concatenating a number of repeating units, such as words or morphemes, *predictability of next (or previous) letter is higher (low entropy) within the units, lower (high entropy) between the units*. Besides linguistic data, this happens to be valid for a wide variety of naturally occurring streams (Cohen et al. 2007). Similar ideas have also been exploited by computational models of human language acquisition. *Simple recurrent networks* used for learning a large variety of linguistic phenomena are trained to guess the next unit in the input sequence, and the networks function based on how predictable the next unit in the test input is. Brent (1996) used *distributional regularities* where a segmentation decision is made at points with unexpected sequences of phonemes. And, indeed, children have been shown to be sensitive to this type of information in the continuous speech stream very early in life (Saffran et al. 1996).

The subject of this paper, the SV method, does not utilize all possible information that can be useful for segmentation task. There are several other sources of information, or cues, that can facilitate the task of learning segmentation. When available, existing *lexical knowledge*; the *words encountered in isolation*; *prosodic cues*, such as word stress; *phonotactic constraints*, such as the phoneme sequences that do not occur beginning or end of the words; *allophonic differences*; and *vowel harmony* are known to be useful for segmentation. However, some of these are not always available, some require a working knowledge of lexical items of the language to be useful. On the other hand, the SV (and similar methods) uses the relationship between the successive basic units, and it is applicable as long as basic units can be discriminated.

We refer to the methods listed so far as *local* methods. In local methods, the segmentation decision is made only by checking the immediate context. Local methods contrast with a large number of successful segmentation algorithms that are based on optimizing a *global* objective function. Global optimization based methods typically rely on two competing factors: one preferring the smaller units, hence segmentation; the other not allowing over-segmentation. The ‘best’

segmentation is, then, the one found by optimizing the combination of these factors. Examples of these include models based on the minimum description length (MDL) (Goldsmith 2001, Goldsmith 2006), maximum a posteriori (MAP) estimate (Creutz and Lagus 2007), or explicit probabilistic (Bayesian) models (Brent 1999, Goldwater et al. 2009).

Most recent approaches to segmentation make use of a number of available cues, and sometimes, a combination of local and global methods. This paper focuses on only one of the local methods, the SV, and suggests an improvement to this method as well as provides an in-depth empirical analysis. The next section will introduce the method. After reviewing related studies in Section 3, we will demonstrate the use of the method on real world data in Section 4. Section 5 describes the suggested improvement. We present a number of experiments demonstrating the effectiveness of the improvement on large word-lists for English and Turkish in Section 6. Section 7 concludes after a brief discussion of the results and other possible improvements.

2 Successor Variety

The successor variety of a string is the number of distinct phonemes (or letters) that can follow the string in the language we are interested in. Harris' use of successor variety was aimed at finding morphemes in spoken language. Given an initial segment of an utterance, a high number of phonemes that may follow the segment was used as an indication of a morpheme boundary. His definition of 'high' number of phonemes depended on human judgments.

We define successor variety as the number of distinct letters[1] that follow a given initial word segment as counted in a word list. More formally, for a suffix[2] x, a letter y and a word list W formed by letters in alphabet A,

$$SV(x) = \sum_{y \in A} c(x, y)$$

where,

$$c(x, y) = \begin{cases} 1 & \text{if string } xy \text{ occurs as an initial word segment in W} \\ 0 & \text{otherwise} \end{cases}$$

For example, given the word list in Figure 1a, and the test word `reading`, Figure 1b presents successor values for the test word. Note that we assumed a hypothetical end-of-word letter after the word `read`. For this small list, the successor values (SV) after `rea-` and `read-` are higher (2 and 3 respectively), potentially indicating morpheme boundaries.

[1] The method and the improvement we suggest are applicable to other units. Our definition is based on letters, as we will only use written language data in this study.

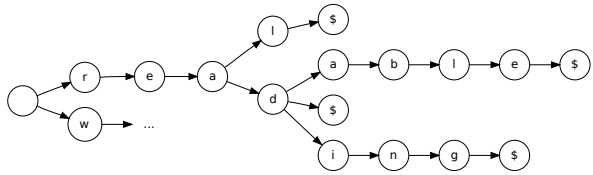
[2] Our use of the terms 'prefix' and 'suffix' in this article refers to any initial or final word segment. These segments are not necessarily linguistic units, i.e., morphemes attached to the beginning or the end of the words. We will clearly state it when these terms are used to refer to linguistic units.

`read`	`readable`	`reading`	`real`
`write`	`writing`	`writer`	`working`

(a)

letter	**r**	**e**	**a**	**d**	**i**	**n**	**g**	
SV	2	1	1	2	3	1	1	1
PV	1	1	1	1	3	1	1	5

(b)



(c)

Figure 1: (a) An example word list. (b) the successor and predecessor values for the test word `reading`. (c) A part of the trie structure representing the given word list. The character '$' represents the hypothetical end-of-word character.

A straightforward extension of the SV is the *predecessor value* (PV), the number of different letters that can precede a certain suffix. The line labeled 'PV' in Figure 1b lists the predecessor values for our example.

The successor and predecessor values for a given word list can easily be computed making use of a data structure called *prefix tree*, or *trie*. If the input word list is inserted into a trie, the successor value for a given prefix is the number of branches after the corresponding node. Figure 1c presents part of the trie for our example.

Intuitively, the successor and predecessor values seem to be simple and usable indications of morpheme boundaries. However, how can we use them to actually find the morpheme boundaries? How can we avoid finding non-morphemes like `rea` in our test word list, and not miss the real morpheme `write`? Given our example word list, `write` has the same SV and PV as the non-morpheme `rea`. What changes with the size of the word list, or language? After reviewing how others dealt with these questions, the rest of the paper tries to fill some of the gaps in answers to these questions.

3 Related Work

Most of the studies in the literature are 'consumers' of the SV based methods (e.g., Déjean 1998, Al-Shalabi et al. 2005, Goldsmith 2006, Demberg 2007, Stein and

Potthast 2008)). All these studies use the measure in one way or another to find morpheme boundaries in words. To our knowledge, there are only two studies that try to analyze and improve the method. The most elaborate study of various SV options is by Hafer and Weiss (1974), and the work by Bordag (2007) combines the SV with other methods to increase its effectiveness.

The goal of Hafer and Weiss (1974) (H&W) is stemming for information retrieval purposes. However, a good part of the paper investigates various options to segment words into morphemes including some variations of the SV method. H&W ran 15 different experiments on English and reported results from 13 of them. The criteria to segment in their experiments depended on a combination of SV and/or PV together with corresponding thresholds (or cutoff values); SV or PV being on a peak or plateau; the suffix or prefix at a certain location being a word by itself; and analogous to SV and PV, *successor entropy* (SE) and *predecessor entropy* (PE). In this paper, we will only focus on the variety scores. However, we will give a brief description of the entropy method, since the entropy scores may be more suitable for certain applications (where precision is more favorable to recall), and the normalization approach we advocate in this paper is also applicable to entropy values.

The segmentation entropy of a given prefix x is defined as:

$$H(x) = -\sum_{y \in succ(x)} \frac{f(xy)}{f(x)}\, log_2 \frac{f(xy)}{f(x)}$$

where, $f()$ returns the frequency (count) of the words starting with the given prefix, xy is the string that is formed by concatenation of string x and the letter y, and $succ()$ returns all successor letters for the given prefix. This formula gives the SE. One can easily obtain PE by modifying $successor$ with $predecessor$ and changing the order of the concatenation.

As H&W also mention, compared with the SV values, the entropy values provide an averaging effect that may reduce noise. However, it also reduces the sensitivity to the real boundaries. Hence, one expects better accuracy from entropy values, while the SV values provide better coverage. To clarify the difference between the entropy and the SV values, consider two hypothetical cases: a particular prefix is followed by (i) 2 different suffixes each starting with different letters, (ii) 5 different suffixes where 4 of them share the same first letter. For both cases SV will be 2, however, SE will be higher for case (i) compared to case (ii).

The best performing methods chosen by H&W use a complex combination of threshold values for variety or entropy scores with the additional knowledge on whether parts of the word occur in the word list as a standalone word or not. Two best performing methods selected by H&W are given below:

- (PW *and* $\text{PV}_i > 5$) *or* ($\text{PV}_i > 17$ *and* $\text{SV}_i > 2$)
- (($\text{SE}_{i-1} = 0$) *and* ($\text{SE}_i > 0.8$ *or* $\text{PE}_i > 0.8$)) *or*
  (($\text{SE}_{i-1} \neq 0$) *and* ((PW *and* $\text{PE}_i > 0.8$) *or* ($\text{PE}_i > 3.0$ *and* $\text{SE}_i > 1.0$))).

where, 'PW' stands for 'prefix is a standalone word' in the word list.

letter	**r**		**e**		**a**		**d**		**i**		**n**		**g**	
SV	27	16		27		10		10		2		2		4
PV	7		5		16		10		25		5		13	28

Figure 2: The SV and PV values for the word `reading` calculated from the CELEX database.

The criteria are complicated. However, they seem to work well, and surprisingly, Al-Shalabi et al. (2005) report that the same criteria work also well for Arabic, a language with different morphological properties.

The most important problems with these criteria are their complexity and the high number of tunable parameters. Even though they seem to also work in different languages (this will be discussed further in Section 4), it is likely that the parameters are language (or even corpus) dependent.

Bordag (2007) uses a different method to improve the SV based measures. Finding morpheme boundaries in the example we have presented in Figure 1 is relatively easy due to the choice of words. As we will present later, finding boundaries on the SV values calculated on a large collection of unrelated words is more difficult. Realizing that, Bordag first does a contextual similarity analysis, and finds a relatively small number (e.g., 150) of words with high probability of semantic relatedness to the target word. The SV values are calculated on this smaller set of words, and similar to H&W, segmentation decision is based on thresholds.

Besides the thresholds for SV or PV values, Bordag's method introduces a new parameter, the number of related words. Additionally, the method's success depends highly on the contextual similarity analysis, and the corpora used for the similarity analysis.

In this study, we introduce a new approach to improve the SV based methods. We propose a method to normalize the SV scores that increases the effectiveness of the method, and allows for an easier interpretation of the scores. However, before introducing the normalization procedure, we will have a closer look at the SV values calculated on real-world data.

4 Successor Variety in Real World Data: a Closer Look

So far, the examples we used were toy examples demonstrating the method. However, the language data in the real world do not come in neatly organized small packages of word lists. In this section we will try to demonstrate some of the general characteristics of the SV measures as seen in real-world examples.

The experiments reported in this paper have been done on two typologically different languages: English and Turkish. For the English data we used the CELEX database (Baayen et al. 1995). As CELEX does not provide standard surface form segmentations that we need for evaluating our methods, we used the segmentations provided by Hutmegs (Creutz and Lindén 2004). The Turkish word list is extracted from the METU Corpus (Say et al. 2002). The Turkish

gold-standard is obtained with the help of a finite-state morphological analyzer (Çöltekin 2010). The number of word types in the English data set is 114184 and the number of words in the Turkish data set is 143275. For the experiments, where using the same number of words is important, we randomly removed 29091 words with frequency one from the Turkish word list.

To get a first idea, we repeat the Figure 1b in Figure 2 this time calculating the values from the complete CELEX word-form list. Predecessor value peaks after `-ing` (traversing right to left). However, we do not see the same on successor values. The only peak value we identify with successor values is after `re-` which is not desirable for this word, but `re-` being a common prefix in English, this is an expected result. If we were to segment successor and predecessor peak values, we would (mistakenly) segment the word as `re-ading`.

The SV and PV values in Figure 2 also demonstrate a general trend: the SV values tend to be higher at the beginning and PV values tend to be higher at the end of the words, and they decrease as they go right and left respectively. To visualize this trend we plotted the mean SV and PV values in Figure 3. In addition to the average SV and PV for the CELEX and METU word lists, Figure 3 also presents a random baseline that we will explain in detail in Section 5. Clearly, the SV values are high at the beginning and the PV values are high at the end of the word (note that in the PV graph the index values are reversed). Except for the separation on the x-axis due to the average word length, and the height of the graphs (partially due to the size of the alphabet), the graphs are rather similar.[3]

Figure 3 indicates a clear problem with the strategies using threshold values and peaks for segmentation. Due to the exponential drop of the SV (and PV), it is difficult to find threshold values that would work everywhere. As the SV values are naturally high at the beginning of the words, a small threshold will suggest incorrect boundaries at the beginning of the words. If the threshold is tuned not to make mistakes at the beginning of the words, then the threshold value will be too conservative for the rest of the word.

The problem affects the performance at the beginning (for SV) and end (for PV) of the words. Unfortunately, most suffixes and prefixes in natural languages are rather short and attach to the end or beginning of the words. Hence, not accounting for these tendencies will make SVs only useful for detecting suffixes, and PVs for prefixes, where otherwise combination of both values may yield a better classification. Luckily, the problem can at least partially be fixed by collecting some simple statistics over the data. The next section will present the proposed solution for this problem and the effectiveness of this solution.

Figure 3 also reveals a difference between the distribution of the SV and PV

[3]Both graphs for Turkish show a clear raise after position 2, causing a peak at 3. This is due to a phenomenon we briefly discuss in Section 7. Turkish has 21 consonants, and 8 vowels. In the Turkish word list, 80% of the first letters are consonants. The probability of seeing a vowel after a consonant is 0.84. Hence, it is more likely to see a vowel as the second letter. Since there are only 8 vowels, as with after any consonant, the SV values after the first letter tend to be low. Similarly, the SV values after the second letter, as with after any vowel, tend to be high. The distribution of the letters at the end of the words, hence the PV values, also follow a similar pattern. For English, having more consonant-consonant and vowel-vowel bigrams, the same effect is observed in the graphs only as a slope change.

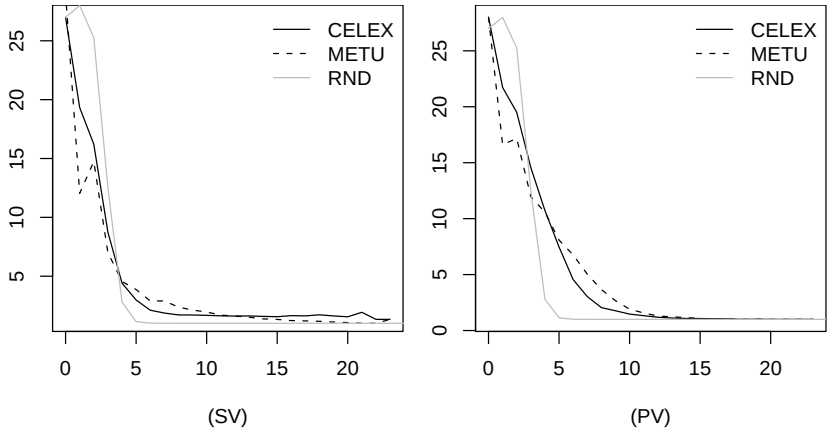


Figure 3: Mean SV and PV values for each word position. Indexes start from the beginning of the word for SV, and from the end of the word for PV.

values. The drop in the SV values is sharper. This is due to the fact that both languages in our study are primarily 'suffixing' languages: except a few productive linguistic prefixes, most of the productive morphological processes are due to the use of suffixes. This is clearly visible in the difference between the drop of the SV values from the beginning of the words to the end and the PV values from the end of the words to the beginning. Even though the languages differ in their morphological productivity, the graphs presented in Figure 3 do not reveal a big difference between the languages. However, we observe a clear difference between the behavior of SV and PV values.

The visualization in Figure 3 also clarifies the unexpected success of the SV thresholds tuned by Hafer and Weiss (1974) across different languages. The successful segmentation criteria they use depend on a combination of successor and predecessor values, where one of the thresholds is high and the other is low. As shown in Figure 3 the SV values tend to be one towards the end of a medium length word, likewise for the PV values towards the beginning. On the other hand, high SV scores can be found at the beginning, and high PV scores can be found at the end of the words. Hence, a high SV threshold in combination with a low (but greater than one) PV threshold translates to 'any PV value greater than one towards the beginning of a word'. And the reverse condition —high PV threshold, low SV threshold— detects any SV value greater than one towards the end of the words.

5 Normalization

The tendency of average SV and PV values dropping exponentially in Figure 3 can partially be explained by a very general process. Any process generating a large

letter	**r**		**e**	**a**	**d**	**i**	**n**	**g**
SV	27	16	27	10	10	3	2	4
PV	7	5	16	10	25	5	13	28
SV/AVG	1.00	0.83	1.67	1.15	2.27	0.68	0.95	2.15
PV/AVG	2.44	1.14	2.18	0.94	1.73	0.26	0.70	1.00

Figure 4: Normalized SV and PV values.

number of word-like strings from a fixed alphabet would generate similar successor and predecessor counts. To demonstrate this, we will consider a process that creates random word like units, and compare it to actual natural language words. The light-gray line in Figure 3 shows the SV and PV values for such a process. We have generated random word-like units from an alphabet of size 30 (approximately the alphabet size for both of our word-lists), and generated $114,184$ (size of the CELEX word list) random 'words'.[4] Even though the random words were not formed by concatenating morphemes, they show the same tendency: At the beginning of the words the SV is high, and at the end of the words the PV is high. Naturally, the drop of the SV values of the random process does not show any differences from the PV values.

The main difference between real language data and the randomly generated data is that, rather than being a random collection of letters, the words in the real language data are formed by concatenating more basic units, morphemes. However, we observe the exponential drop of the SV and PV values for both the randomly generated data and the real language words. Hence, removing the effect of the letter concatenation process from the SV and PV values calculated for the real data should reveal the underlying process of morpheme concatenation better. To do that, we will follow a simple method: we will divide each variety value by the expected value in that position.[5]

Along with the previous SV and PV values presented in Figure 2, Figure 4 presents the normalized scores. The first difference to note is that both normalized scores for successor and predecessor values peak after `read-`. Since an SV value of 10 is closer to the expected value in position 3 (after `rea-`) than in position 4 (after `read-`) the scores are more sensitive to real boundaries. Of course, this may also increase the number of potential false positives. However, the results we present in Section 6 show that the normalization is indeed beneficial for increasing

[4] During the random word generation, letters are sampled similar to the letter distribution in the CELEX word list, and the word length distribution has been estimated from the combined 11 languages from the Europarl corpus. While providing a language-neutral word length distribution, this results in relatively long words compared to the more complete/balanced CELEX and METU word lists.

[5] Subtracting the expected SV/PV values from the calculated ones is another, arguably more intuitive, approach to normalize the SV values. The values obtained by subtracting the expected value from the calculated values have a similar distribution with the log of the values obtained by the normalization by division. The subtraction method does not require further log transformation, and it leads to higher accuracy in some of the conditions we tested. However, the subtraction method is more prone to changes in the size of the word list. The main reason for choosing to normalize by dividing is to get properly scaled values for the (semi-)supervised experiments discussed in the next section.

the segmentation performance.

The performance increase will be more apparent with the empirical tests. However, we can see another benefit of the normalization by looking at our example, and by visualizing the data. Intuitively, a normalized score of one means that the variety score is the same as the expected value. A score less than one is below the expected value, and a score greater than one is higher than the expected value. Hence, regardless of the position in the word, if the value is less than or around one there is no reason to get surprised, and no need to posit a boundary. However, if the value is significantly greater than one, it is more likely to be a boundary. And if we we take an even closer look at the data, we can see that the normalized variety values are approximately distributed according to a log-normal distribution. As a result if we take the normalization one step further, and get the logarithm of the values, we end up working with approximately normally distributed values. Figure 5 presents plots of density estimates of normalized and unnormalized SV and PV scores for both boundary (black lines) and non-boundary (gray lines) positions for English and Turkish word lists. In all cases, the modes of the distributions of boundaries are greater than the modes of the non-boundaries. However, Figure 5 also demonstrates that, for the SV values, the overlap between boundary and non-boundary values is clearly higher for the non-normalized case, and separation is not as clear as in the normalized case. Unfortunately, we do not see the same positive effect of normalization on the PV values. This is because of the fact that both languages are primarily suffixing languages and the suffixes tend to be shorter than the stems. The PV values at the end of the words are higher because of the process of morpheme concatenation as well. As a result, it is not possible to see if the high number of PV values at the end of a word is because of a genuine morpheme boundary, or because of the letter concatenation process. Since the number of genuine morpheme boundaries is lower at the beginning of the words, the SV values do not suffer from this phenomenon in either language we studied.

For both the SV and PV, the log-normalized values are (approximately) normally distributed. By estimating the parameters of the normal distributions for boundaries and non-boundaries, we can easily come up with strategies based on the SV that best suit the application at hand. A neutral method would be to view the segmentation task as binary classification of the data coming from two Gaussian distributions. Similarly, a more conservative estimate can be obtained by segmenting at values that are at the right tail of the combined normal distribution. For ease of comparison, in our experiments we will tune a threshold value that generates the best F_1-score for both normalized and unnormalized cases.

Another point to note from the visualization of the data in Figure 5 is that even though the unnormalized distributions for different languages seem to be rather different, the normalization takes away this dissimilarity. This may allow us to use strategies that are relatively language neutral. That is, by using normalized scores, the same model may work better cross-linguistically.

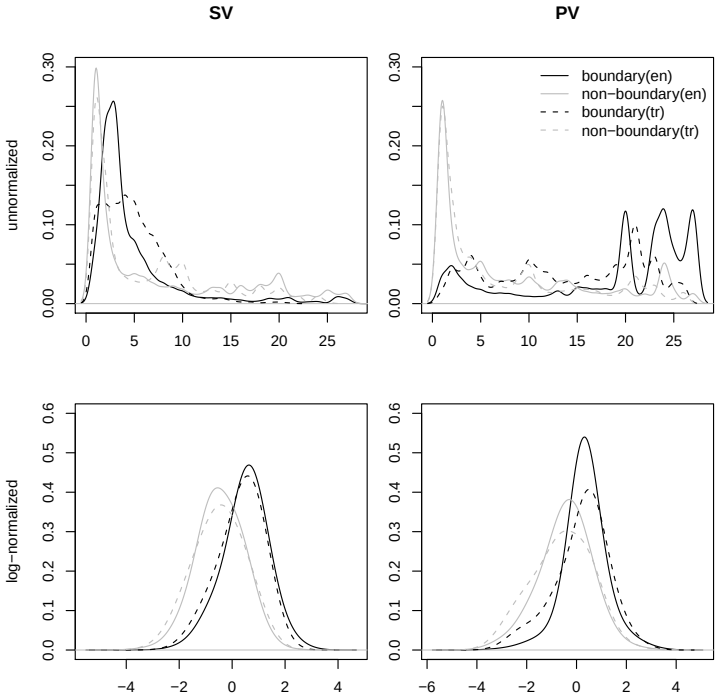


Figure 5: Density estimates for the SV and PV values with or without normalization.

6 Experiments

The most common application of using successor values for morphological segmentation has been in unsupervised morphological segmentation and analysis tasks. However, in all the examples in the literature that we are aware of, the method depends on thresholds tuned on a specific word list or intuitively by the designers of the algorithms. Arguably, the completely unsupervised use of the method is using changes (peaks) in the value of SV and PV as a criterion for segmentation. However, a small threshold is generally useful to guard against over-segmentation, and the methods deciding on the basis of a threshold tend to perform better. Another approach for completely unsupervised learning is based on taking the normalized SV and PV values as coming from two different (normal) distributions, and inducing the parameters of these distributions using an unsupervised method. Although complete unsupervised methods are worth exploring for their own sake, the methods we present below provide a better comparison between the normalized and unnormalized values, as well as comparing them to the previous results found in the literature.

To be able to demonstrate the effects of the normalization on the performance of the segmentation task, we present two sets of experiments in this section. First, we will present the *precision*, *recall* and F_1*-score* values for the segmentation of English and Turkish data sets using the SV and PV values individually, using a peak criterion on SV or PV values (SP an PP) and simple logical *and* and *or* combination of these values. In the second set of experiments, we will train a simple linear classifier.

In the first set of experiments we found threshold values that produce the best average F_1-score for 10-fold cross validation on our data sets. First we used thresholds for the individual SV and PV values. Second, we used the 'peak criterion', where we assumed a boundary where the value shows an increase. And last, we used simple logical *and* and *or* combinations of the SV and PV values.

Table 1.1 presents these results, along with two baselines. The line marked as 'Letters' presents the simple strategy of segmenting words to single letters. The line marked as 'Morfessor' presents the results obtained using the Morfessor 1.0 baseline (Creutz and Lagus 2005). As expected, by using the normalized SV values, we can find thresholds that perform better. However, the normalization for PV values produce even worse than non-normalized values, which also took F_1-scores for some of the combinations down. This can be corrected by adding another criterion by favoring boundaries at the end of the words. This is indeed useful for improving the performance of the normalized PV based segmentation criteria in Table 1.1. This type of correction is rather ad hoc and introduces additional parameters. For the simple experiments, we will not report the results of such an ad hoc correction here. However, the effect of using the information on the position in the word is demonstrated in the second set of experiments we report below.

We did a large number of tests and presented in Table 1.1 to get an insight into the method, and to be able to compare the effectiveness of the measures. However, tuning thresholds and combining multiple indicators in a sensible way is, at best,

		English			Turkish		
	Method	P	R	F_1	P	R	F_1
Letters		0.19	1.00	0.32	0.22	1.00	0.36
Morfessor		0.82	0.55	0.66	0.77	0.50	0.60
unnormalized	SV	0.25	0.93	0.40	0.34	0.74	0.47
	PV	0.51	0.65	0.57	0.42	0.78	0.55
	SP	0.46	0.59	0.52	0.47	0.49	0.48
	PP	0.54	0.52	0.53	0.37	0.38	0.38
	SV and PV	0.65	0.69	0.67	0.62	0.66	0.64
	SV or PV	0.40	0.67	0.51	0.38	0.90	0.53
normalized	SV	0.42	0.72	0.53	0.45	0.74	0.56
	PV	0.34	0.88	0.49	0.44	0.68	0.53
	SP	0.53	0.59	0.56	0.37	0.78	0.50
	PP	0.36	0.60	0.45	0.31	0.75	0.44
	SV and PV	0.61	0.70	0.65	0.67	0.57	0.62
	SV or PV	0.42	0.73	0.53	0.45	0.74	0.56

Table 1.1: *P*recision/*R*ecall/*F*-score values optimized for best F-score. The first block presents two baselines.

a cumbersome task. A better way to make use of these values is using them as features in a (semi-)supervised learning method. The second set of experiments we conducted is based on training a simple linear classifier (Fan et al. 2008) using the normalized and unnormalized SV and PV values. As well as the SV and PV values, we used two other sets of features. First, two additional binary features that indicate the occurrence of the suffix and the prefix (in respect to the candidate boundary) as a separate word in the word list. Second, two index numbers corresponding to the offsets of the candidate boundary from the beginning and end of the word. The standalone occurrence of suffix and prefix in the word list is frequently used by the other models in the literature. The addition of index features, on the other hand, allows the learner to correct the problem with the normalized PV values that we observed above.

The results of the supervised learning experiments are presented in Table 1.2. Each row in the table presents the precision, recall and F_1-Scores for different combination of the features. All values are averages of 10-fold cross validation on the CELEX and METU word lists. The experiments reported at the rows marked with '+index' have been generated using two integer features representing the index from the beginning and the end of the word. The rows marked '+w' report the results from the experiments where we added two additional binary features indicating the existence of the suffix and the prefix as a standalone word in the word list. Except the 'PV' and 'SV+PV' features for English, all combinations of the features show a consistent increase of F_1-score for both languages. And as expected, the addition of '+index' also allows the model to generalize better using

		English			Turkish		
	Method	P	R	F_1	P	R	F_1
unnormalized	SV	0.51	0.35	0.41	0.48	0.55	0.51
	PV	0.29	0.64	0.40	0.27	0.91	0.42
	SV+PV	0.40	0.84	0.54	0.47	0.71	0.57
	SV+PV+index	0.40	0.89	0.55	0.40	0.85	0.55
	SV+PV+index+w	0.46	0.83	0.59	0.46	0.87	0.60
normalized	SV	0.36	0.83	0.50	0.45	0.75	0.56
	PV	0.30	0.69	0.42	0.45	0.69	0.55
	SV+PV	0.39	0.85	0.53	0.56	0.76	0.64
	SV+PV+index	0.43	0.83	0.57	0.59	0.80	0.68
	SV+PV+index+w	0.60	0.83	0.69	0.62	0.81	0.70

Table 1.2: Results of supervised learning.

SV and PV values.

Encouraged by the similarity of normalized distributions for two languages in Figure 5, we conducted four more experiments: we trained the classifier using the full set of features on one of the languages, and tested on the other. As expected, the models trained by the unnormalized data performed poorly. Training on Turkish data and testing on English data resulted in an F_1-score of 0.34, barely above the 'Letters' baseline. Training the system with English and testing on Turkish did even worse, an F_1-score of 0.23. However, the F_1-scores using normalized scores for the same setup resulted in F_1-scores of 0.67 and 0.64 respectively—better than any of the hand-tuned thresholds in Table 1.1.

7 Discussion and Conclusions

Along with an in-depth analysis, we presented in this paper a method to improve successor variety, an old but frequently used measure for segmentation. The measure we discussed in this paper, successor variety, shares a principle with a few other measures used in the literature: the predictability within the units (e.g., morphemes) is high, predictability between the units is low. The analysis presented here focuses on the application of the successor variety to segment words into morphemes. However, the improvement suggested in this paper can be used in other segmentation applications, and can be applied to other measures, e.g., entropy, if used in a similar fashion.

The normalization idea presented here is based on the observation that, even for a random letter concatenation process, the successor values tend to be high at the beginning, and drop exponentially as we increase the prefix length. This suggests that if we can isolate the concatenation at a higher level—in our case concatenation of the morphemes— from the letter concatenation process, we can increase the efficiency and arrive at a better interpretation of the relation between the measure

and the boundaries. The normalization method we presented in this paper achieves that by simply dividing the calculated successor value to the expected value after an equal prefix length. Alternative normalization methods are possible, for example simply subtracting the expected value from the calculated score.

While calculating the expected value, our method only considers the length of the prefix (the expected SV is higher for shorter prefixes). However, natural languages have a number of other regularities that affect the SV values. Particularly, not every letter or letter class is likely to be followed by equal number of successor letters. For example, the languages we consider have more consonants than vowels, and typically, consonants are followed by vowels and vowels are followed by consonants. This results in vowels in average to have higher successor values than consonants. Arguably, incorporating this information while calculating the expected values may provide a better normalization. In a number of preliminary experiments that we did not report in this paper, we could not find any consistent improvement by incorporating this information in the normalization process.

The experiments we conducted in two different languages demonstrate that the normalization method proposed here is effective in increasing the performance of the segmentation methods based on successor variety. The effect seems to be more useful for the SV values, however, with correct use of other cues, we also demonstrated that it may increase the effectiveness of the PV values as well.

References

Al-Shalabi, Riyad, Ghassan Kannan, Iyad Hilat, Ahmad Ababneh, and Ahmad Al-Zubi (2005), Experiments with the successor variety algorithm using the cutoff and entropy methods, *Information Technology Journal*.

Baayen, R. Harald, Richard Piepenbrock, and Léon Gulikers (1995), The CELEX lexical database (CD-ROM).

Bordag, Stefan (2005), Unsupervised knowledge-free morpheme boundary detection., *The Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*.

Bordag, Stefan (2007), Unsupervised and knowledge-free morpheme segmentation and analysis, *The Working Notes for the CLEF Workshop 2007*.

Brent, Michael R. (1996), Advances in the computational study of language acquisition, *Cognition* **61**, pp. 1–38.

Brent, Michael R. (1999), An efficient, probabilistically sound algorithm for segmentation and word discovery, *Machine Learning* **34** (1–3), pp. 71–105.

Cohen, Paul, Niall Adams, and Brent Heeringa (2007), Voting experts: An unsupervised algorithm for segmenting sequences, *Intelligent Data Analysis* **11** (6), pp. 607–625.

Çöltekin, Çağrı (2010), A freely available morphological analyzer for turkish, *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC2010)*.

Creutz, Mathias and Krista Lagus (2005), Inducing the morphological lexicon of a natural language from unannotated text, *Proceedings of the International*

and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05), Espoo, Finland, pp. 106–113.

Creutz, Mathias and Krista Lagus (2007), Unsupervised models for morpheme segmentation and morphology learning, *ACM Trans. Speech Lang. Process.* **4** (1), pp. 3, ACM, New York, NY, USA.

Creutz, Mathias and Krister Lindén (2004), Morpheme segmentation gold standards for Finnish and English., *Publications in Computer and Information Science A77*, Helsinki University of Technology.

Déjean, Hervé (1998), Morphemes as necessary concept for structures discovery from untagged corpora, *Workshop on Paradigms and Grounding in Natural Language Learning*, pp. 295–299.

Demberg, Vera (2007), A language-independent unsupervised model for morphological segmentation, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Association for Computational Linguistics, Prague, Czech Republic, pp. 920–927.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin (2008), Liblinear: A library for large linear classification, *Journal of Machine Learning Research* (9), pp. 1871–1874.

Goldsmith, John (2001), Unsupervised learning of the morphology of a natural language, *Computational Linguistics* **27** (2), pp. 153–198, MIT Press, Cambridge, MA, USA.

Goldsmith, John (2006), An algorithm for the unsupervised learning of morphology, *Natural Language Engineering* **12** (04), pp. 353–371.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson (2009), A Bayesian framework for word segmentation: Exploring the effects of context, *Cognition* **112**, pp. 21–54.

Hafer, Margaret A. and Stephen F. Weiss (1974), Word segmentation by letter successor varieties, *Information Storage and Retrieval* **10** (11–12), pp. 371–385.

Harris, Zellig S. (1955), From phoneme to morpheme, *Language* **31** (2), pp. 190–222, Linguistic Society of America.

Saffran, Jenny R., Richard N. Aslin, and Elissa L. Newport (1996), Statistical learning by 8-month old infants, *Science* **274** (5294), pp. 1926–1928.

Say, Bilge, Deniz Zeyrek, Kemal Oflazer, and Umut Özge (2002), Development of a corpus and a treebank for present-day written Turkish, *Proceedings of the Eleventh International Conference of Turkish Linguistics*.

Stein, Benno and Martin Potthast (2008), Putting successor variety stemming to work, *in* Decker, Reinhold and Hans J. Lenz, editors, *Advances in Data Analysis*, Springer, pp. 367–374.

Dutch Named Entity Recognition using Classifier Ensembles

Bart Desmet and Véronique Hoste

LT3, Language and Translation Technology Team, University College Ghent
Department of Applied Mathematics and Computer Science, Ghent University

Abstract

This paper explores the use of classifier ensembles for the task of named entity recognition (NER) on a Dutch dataset. Classifiers from 3 classification frameworks, namely memory-based learning (MBL), conditional random fields (CRF) and support vector machines (SVM), were trained on 8 different feature sets to create a pool of classifiers from which an ensemble could be built. A genetic algorithm approach was used to find the optimal ensemble combination, given various voting mechanisms for combining classifier outputs. The experiments yielded a classifier ensemble that outperformed the best individual classifier by 0.67 percentage points (F-score), a small but statistically significant margin. Experimental results also showed that ensembling classifiers from different frameworks benefits generalization performance.

1 Introduction

Named Entity Recognition (NER) is the task of automatically identifying names within text and classifying them into categories, such as persons, locations and organizations. NER started as an information extraction subtask, but has since evolved into a distinct task essential for information retrieval, question answering, and as a preprocessing step for coreference resolution and various other NLP problems.

An extensive literature on the subject exists (Chinchor 1998, Tjong Kim Sang and De Meulder 2003, Cucerzan 2007), with NER approaches roughly falling into three categories: hand-crafted, machine learning and hybrid systems. Hand-crafted approaches require manual rule creation, a time-consuming process which hinders easy porting to new domains or languages. Supervised machine learning solutions, on the other hand, rely on an annotated training corpus to infer patterns associated with named entities, based on morphological, syntactic, lexical and contextual features. Hybrid systems combine both approaches.

For machine learning systems, named entity recognition is usually regarded as a classification task. In a two-step approach, each token in a text first has to be classified as either belonging to a named entity chunk or not (named entity recognition), and afterwards, the chunks labeled as named entities are classified according to type (named entity classification). One-step NER combines both steps by classifying each token either as one of the named entity types or as *not-a-named-entity*.

A variety of machine learning algorithms has been applied to the NER task. Research is often aimed at finding the most informative features, and discarding

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

the uninformative ones (e.g. Isozaki and Kazawa (2002)), or finding the right settings for a specific algorithm (e.g. De Meulder and Daelemans (2003)). Feature selection and parameter optimization are aimed at improving a single classifier's performance. However, finding the optimal features and parameters is a complex problem.

An alternative research direction is that of combining several classifiers into an ensemble, and combining their output using a voting procedure (e.g. Wang et al. (2008)). The assumption is that combining a diverse set of classifiers improves the generalization accuracy, provided that the ensemble's members have sufficient individual performance and the errors they make are, to some extent, non-overlapping. Again, finding such an ensemble is a non-trivial problem.

The work in this paper closely follows the approach proposed in Ekbal and Saha (2010). Ekbal and Saha describe a system that uses genetic algorithms to find an optimal classifier ensemble. A genetic algorithm is a method to find or approximate solutions to a search problem. The technique is inspired by evolutionary biology, applying evolutionary concepts such as selection, inheritance, mutation and crossover to a population of possible solutions, in order to find the solution that is most fit to the problem (Whitley 1994).

The system by Ekbal and Saha (2010) selected an optimal classifier ensemble from a set of 19 maximum entropy classifiers. They evaluated the ensemble classifiers on Bengali, Hindi, Telugu and English datasets, and report F-score improvements over the best individual classifiers of 5.63, 1.95, 5.75 and 12.88, respectively.

In this paper, we investigate if a similar system can successfully be applied to construct the best classifier ensemble from a set of classifiers from three different classification frameworks, namely memory-based learning, conditional random fields and support vector machines. We evaluate the performance of this ensemble on a Dutch data set, and compare the results to individual classifier performance.

The remainder of this paper will be structured as follows. In Section 2, we present the features used for the classifiers. The three classification frameworks are introduced in Section 3. The dataset, classifier pool, voting mechanisms and genetic algorithm used for the ensemble selection experiments are discussed in Section 4. The results of the experiments are presented and discussed in Section 5, and Section 6 concludes this paper.

2 Feature extraction

Machine learning systems are not directly trained on a corpus. The information present in a corpus and its annotations needs to be translated into a collection of instances, in order for a learning algorithm to infer classifications from them. Each instance represents a subsection of the corpus that carries a meaningful classification (as stored in the annotations).

In the case of named entity recognition, every token from the corpus is represented by an instance, which has a class indicating whether the token is a named

entity, and if so, which type. An instance represents the token by means of a feature vector, which is a list of characteristics of the token and its context that are deemed relevant for the classification task.

We extracted a range of features, many of which are commonly used in the field (Bogers 2004).

Basic information

- the original token
- the POS tag, which was obtained by preprocessing the data with the Memory-Based Shallow Parser (Daelemans and van den Bosch 2005)

First word: A binary feature indicating if the word is in sentence-initial position.

Orthographic information: Non-exclusive binary features capturing the orthographic characteristics of the token, such as capitalization, hyphenation and the occurence of numbers and punctuation marks.

- `firstCap`: Is the first letter capitalized?
- `allCaps`: Is the entire word uppercased?
- `internalCaps`: Does the token contain uppercased letters, apart from the first one?
- `allLowercase`: Is the entire token lowercased?
- `containsDigit`: Does the token contain at least one digit?
- `containsDigitAndAlpha`: Does the token contain at least one digit and one alphanumeric character?
- `onlyDigits`: Is the entire token made up of digits?
- `isHyphenated`: Does the word contain at least one punctuation mark and any other character?
- `isPunctuation`: Does the token only contain punctuation marks?
- `containsPunctuation`: Does the token contain at least one punctuation mark?

Word shape: A symbolic feature that tests for the same orthographic characteristics as the binary features described above, outputting one of the following labels: *allLowercase, allCaps, firstCap, capPeriod, onlyDigits, containsDigitAndAlpha, allCapsAndPunct, firstCapAlphaAndPunct, alphaAndPunct, onlyPunct, mixedCase* or *other*.

Patterns: Binary features indicating whether the token matches a regular expression that tests for a specific word pattern.

- `isInitial`: Does the token resemble an initial? Initials are defined as strings with up to five capitalized letters separated with periods.
- `isURL`: Is the token a URL? URLs are taken to be strings starting with *http*.

Word length: The number of characters in the token.

Affix information

- `prefix3` and `suffix3`: The first and the last 3 characters of the token
- `prefix4` and `suffix4`: The first and the last 4 characters of the token

Function word: A binary feature indicating whether the token occurs in a list of Dutch function words.

Chunks: A symbolic feature with a base phrase chunk tag, obtained with the Memory-Based Shallow Parser.

Class tag: The correct classification is taken from the annotations, and is represented by one of 13 possible class tags, encoded in IOB2 notation (Tjong Kim Sang 2002b): *B-EVE, I-EVE, B-LOC, I-LOC, B-MISC, I-MISC, B-ORG, I-ORG, B-PER, I-PER, B-PRO, I-PRO* for the six named entity types (see 4.1) or *O* if the token is not part of a named entity.

3 Classification frameworks

A variety of machine learning algorithms have successfully been applied to the task of named entity recognition. We hypothesized that ensembling different types of classifiers would benefit the ensemble performance, assuming that each classifier type makes different kinds of errors. We therefore experimented with 1 lazy and 2 greedy learners.

3.1 Memory-based learning

Memory-based learning algorithms are called lazy learners because they perform no generalization on the instance base they are trained on (Daelemans and van den Bosch 2005). All the instances are stored in memory, and new instances are classified by comparing them to the instance base, for example with a *k*-nearest neighbour algorithm. When a *k*-value of 1 is used, the classifier labels an unseen instance with its closest neighbour in the instance base. Various distance and feature weighting metrics can be used to determine which neighbour is closest. For larger values of *k*, some voting mechanism has to be applied to choose one class label from the nearest neighbours set.

We experimented with TiMBL[1], version 6.2.1 (Daelemans et al. 2009). The instances provided to TiMBL were windowed, in order to provide the algorithm

[1] `http://ilk.uvt.nl/timbl/`

with context information. Preliminary experiments showed that a left context of 3 tokens and a right context of 1 token yielded the best results. Although every feature was windowed like this, further experiments should establish for which features windowing is informative.

3.2 Conditional Random Fields

A Conditional Random Field (CRF) is a probabilistic classifier that is used to segment and label sequential data, which makes it especially apt for natural language processing tasks like named entity recognition. CRFs take an input sequence X with its associated features, and try to infer a hidden sequence Y, containing the class labels. They are as such comparable to Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MEMMs). However, CRFs, unlike HMMs, do not assume that all features are independent, and they can take future observations into account using a forward-backward algorithm, unlike MEMMs, thus avoiding two fundamental limitations of those models (Lafferty et al. 2001).

For our experiments, CRF++[2] version 0.53 was used. CRF++ is a sequence tagger, which requires a template file that specifies the combinations of features it needs to consider.

3.3 Support Vector Machines

A Support Vector Machine (SVM) is a learning classifier capable of binary classification. It learns from the training instances by mapping them to a high-dimensional feature space, and constructing a hyperplane along which they can be separated into the two classes. New instances are classified by mapping them to the feature space and assigning a label depending on its position with respect to the hyperplane. SVMs are said to have a robust generalization ability (Vapnik and Cortes 1995).

For multiclass classification problems, separate SVMs have to be built. With the *pairwise* approach, one SVM is trained for every pair of classes. Another method is *one vs rest*, where one SVM is built for each class to distinguish it from all other classes.

The SVM implementation used in our experiments is YamCha[3], version 0.33 (Kudo and Matsumoto 2003), with pairwise multiclass classification. Like TiMBL, SVM uses windowed instances to be informed about its context.

4 Ensemble selection experiments

The aim of this paper was to determine whether genetic algorithms can be used to find an optimal classifier combination that outperforms any individual classifier and the combination of all classifiers in the pool. To that end, a data set was

[2] `http://crfpp.sourceforge.net/`
[3] `http://chasen.org/~taku/software/yamcha/`

selected and used to train a pool of classifiers, which could be combined in an ensemble. Such an ensemble determines the class tag by means of weighted voting. A genetic algorithm was used to find the optimal combination of classifiers.

4.1 Dataset

The dataset used for the experiments is derived from the STEVIN[4]-funded SoNaR corpus[5]. Before SoNaR, the data from the CoNLL-2002 shared task (Tjong Kim Sang 2002a), containing 309,686 words from four editions of the Belgian newspaper "De Morgen" of 2000, constituted the only corpus annotated with named entity information. The SoNaR project consortium aims to produce a 500-million-word reference corpus of written Dutch containing a wide spectrum of genres and text types (Oostdijk et al. 2008). A 1-million-word subset will be provided with a number of manually corrected annotation layers, including four semantic ones: named entities, coreference relations, semantic roles and spatiotemporal expressions (Schuurman et al. 2009). The subset contains these various text types, reflecting the global corpus design. The manually annotated subcorpus will be used to train classifiers for the automatic annotation of the remaining 499 million words.

For the named entity annotation of the corpus, new annotation guidelines were developed, based on the guidelines from MUC-7 (Chinchor and Robinson 1997) and ACE (LDC 2008). A number of adaptations were made, including the addition of separate categories for products (e.g. *iPad*) and events (e.g. *World War II*), apart from the usual categories for persons, organizations, locations and miscellaneous named entities (Desmet and Hoste 2010).

Because annotation of the SoNaR corpus is ongoing, the ensemble selection experiments were run on a subset of the corpus that had been entirely annotated and double-checked. This subcorpus consisted of 99 documents of the same text type, namely autocue scripts for news shows on Dutch public television.

The distribution of named entities in this dataset can be found in Table 2.1.

4.2 Classifier pool

In order to have a diverse pool of classifiers, 8 different feature sets were used to derive instance bases from the data set. The composition of each set is presented in Table 2.2. We attempted to keep the composition similar to the sets used by Ekbal and Saha (2010). The basic features and first word information was included in every feature set, because omitting them yielded classifiers that were deemed too weak for inclusion in an ensemble.

These feature sets were combined with 4 classification configurations, which were found to perform well and reasonably fast with all the features. No parameter tuning was performed, because

[4] http://taalunieversum.org/taal/technologie/stevin/
[5] http://lands.let.ru.nl/projects/SoNaR/

Label	No. of chunks
EVE	256
LOC	6,624
MISC	787
ORG	2,461
PER	3,290
PRO	400
All NEs	13,818
O	188,461
Total	202,279

Table 2.1: Statistics of the dataset.

Set	1	2	3	4	5	6	7	8
Basic	X	X	X	X	X	X	X	X
First word	X	X	X	X	X	X	X	X
Orthographic	X	X	X		X	X	X	X
Word shape	X	X		X	X		X	X
Patterns	X	X		X			X	
Word length	X	X		X			X	
Affix	4	3		4	4	4		4
Function word	X	X		X				X
Chunks	X	X		X	X		X	X

Table 2.2: Composition of the 8 feature sets.

- TiMBL with default settings: the IB1 (k-nearest neighbour) algorithm with a k-value of 1, overlap as the distance metric and gain ratio feature weighting.
- TiMBL with the IB1 algorithm and a k-value of 7, overlap as the distance metric, gain ratio feature weighting and normal majority voting. This second set of TiMBL classifiers was added to have an equal amount of lazy and greedy learners. We only varied the k-value, without changing the default parameters, which already resulted in classifiers that performed very differently from the $k1$-classifiers. They were therefore considered interesting for the classifier pool.
- CRF++ with the standard feature template.
- YamCha, using a pairwise multi-class strategy.

Each combination of a feature set and a configuration was used to classify the dataset, using threefold cross-validation. This resulted in 32 files with class tags for every token, to be used for the ensemble voting procedure. Overall F-scores

on these files were calculated with the `conlleval` script that was used for the CoNLL 2002 shared task (Tjong Kim Sang 2002a), and are reported in Table 2.3.

Set	1	2	3	4	5	6	7	8
TiMBL *k=1*	74.29	74.28	72.13	75.06	75.31	76.59	68.50	74.35
TiMBL *k=7*	70.97	72.15	65.07	71.71	69.85	71.36	66.32	69.74
CRF++	83.76	83.77	79.97	83.72	83.48	83.69	80.49	83.62
YamCha	82.54	82.69	81.43	82.04	83.04	83.23	80.68	82.67

Table 2.3: Overall F-scores for each individual classifier.

Table 2.3 shows that the CRF classifiers present in the pool perform best on average. The TiMBL classifiers with a *k*-value of 7 get the lowest F-scores. The best individual classifier is the CRF classifier trained with feature set 2.

4.3 Voting

When an ensemble of classifiers is used to determine the class of an instance, some sort of voting mechanism is needed to combine the class tags each individual in the ensemble has assigned to that instance. In our experiments, four voting systems were implemented and tested:

- Normal majority voting: every classifier casts a vote for a class tag, and the tag with the highest score wins. In case of a tie, the most frequent class is chosen. This is an unweighted voting system: all classifiers have an equal amount of influence on the outcome of the vote.
- Globally weighted voting: the weight of a classifier's vote is determined by its overall F-score on the dataset. Classifiers that perform well globally thus have a bigger influence in every vote.
- Class weighted voting: a classifier's vote for one particular class is weighted by its F-score on that particular class. The weight of a classifier thus depends on its performance for the class it is voting for.
- Smoothed class weighted voting: the same principle as class weighted voting, but a classifier's F-score per class is divided by the average F-score of all classifiers for that class. This smoothes the difference in weight between a vote cast for a class for which all classifiers perform well and one cast for a class that is harder to predict.

4.4 Genetic Algorithm

The genetic algorithm approach used for the experiments is inspired by the one described in Ekbal and Saha (2010). It was implemented in Python using the Pyevolve framework[6].

[6] `http://pyevolve.sourceforge.net/`

Genetic algorithms operate on a genome, which is a representation of the search space in which an optimal solution needs to be found. The genome for the problem of selecting an optimal ensemble from a set of n classifiers can be a binary string of length n, in which every bit represents a classifier. In our experiments, $n = 32$, so the chromosome 01010101010101010101010101010101 represents an ensemble in which every second classifier is used.

The search space defined by the genome is explored as follows:

1. An initial population *P(0)* is created, containing $|P|$ randomly sampled chromosomes. A population size $|P|$ of 50 was used in our experiments.

2. For each chromosome, a fitness score is calculated. This is done by having the classifier ensemble, as encoded by the chromosome, vote on the class tag of every instance in the dataset, and then calculating the overall F-score over all the class tags using `conlleval`. The closer this F-score is to 100, the fitter the chromosome is.

3. Rank Selection is used to pick the chromosomes that will populate an intermediate population. First, the 50 chromosomes are ranked according to fitness. The least fit chromosome then receives fitness 1, the second least fit chromosome receives fitness 2, and so forth, with the fittest chromosome receiving fitness 50. Afterwards, an intermediate population of size 50 is populated with chromosomes that are sampled with a probability relative to their fitness. The fittest chromosome thus has the highest probability of being sampled multiple times in the population.

4. When selection is complete, recombination on the intermediate population can be performed to create the next generation *P(1)*. This was done using Single Point Crossover (swapping the genetic code of two chromosomes from one point onwards) with a probability of 0.90.

5. Each chromosome had a probability of 0.02 to be mutated using Flip Mutation, giving every bit in the chromosome a 2 percent chance of being flipped from 0 to 1 or vice versa.

6. Steps 2 to 5 are repeated until a predefined number of generations has been evaluated. We stopped the evolution after 40 generations. The individual with the highest fitness score in *P(40)* is considered the optimal classifier ensemble found by the GA.

The selection and mutation types and probabilities are Pyevolve's default parameters. We used the same population size and number of generations as used in the experiments described in Ekbal and Saha (2010).

5 Results and discussion

The genetic algorithm was run on the dataset with the 4 voting mechanisms described in Section 4.3. Table 2.4 presents the best-performing classifier ensembles

per voting mechanism. The precision, recall and F-scores of these ensembles, the ensembling of all classifiers and the best individual classifier are presented in Table 2.5.

Voting mechanism	Genome
Normal majority	00010100 00000000 11001001 01000100
Globally weighted	00000100 00000000 01010001 01001100
Class weighted	00010100 00000000 11011000 01001100
Smoothed class weighted	00010100 00000000 01011101 11000100

Table 2.4: Best-performing classifier ensembles per voting mechanism. The first 8 bits represent the TiMBL *k=1* classifiers, ordered per feature set, followed by 8 TiMBL *k=7*, 8 CRF++ en 8 YamCha classifiers.

	Precision	Recall	F-score
Ensemble selected by GA			
Normal majority	85.12	83.77	84.44
Globally weighted	85.24	83.61	84.41
Class weighted	84.99	83.36	84.17
Smoothed class weighted	85.32	83.47	84.38
Ensembling of all classifiers			
Normal majority	82.49	82.09	82.29
Globally weighted	82.87	82.21	82.54
Class weighted	82.44	81.63	82.03
Smoothed class weighted	82.59	82.00	82.29
Best individual classifier	84.83	82.73	83.77

Table 2.5: Overall precision, recall and F-scores.

A first observation that can be made about the results presented in Table 2.5 is that the type of voting mechanism, used for combining the class tags of each individual classifier in an ensemble, does not appear to have much influence on the performance of the best classifier ensemble found by the genetic algorithm. The differences in F-scores between the voting mechanisms used with all the classifiers are somewhat more outspoken, with globally weighted voting and class weighted voting yielding the best and worst results, respectively.

It can also be observed in Table 2.4 that the best-performing classifier ensembles, regardless of the voting mechamism used, consist of classifiers from all three classification frameworks, although none of the TiMBL classifiers with k=7 is used. Especially interesting is the occurrence of the TiMBL classifiers with k=1 trained on feature sets 4 and 6, present in one but all and all classifier ensembles, respectively. These classifiers achieve an individual F-score of 75.06 and 76.59,

respectively, well below the F-scores of the selected CRF and SVM classifiers. This observation may corroborate that combining different types of learning algorithms in a classifier ensemble can lead to better generalization performance of an ensemble.

All best-performing classifier ensembles outperform the ensembles consisting of all classifiers by a significant margin. The difference in F-score between the best-performing classifier ensemble (normal majority voting, 84.44) and the best-performing individual classifier (CRF++ trained on feature set 2, 83.77) is 0.67 percentage points. This difference was found to be statistically significant.

For the calculation of statistical significance of the F-score, we applied the bootstrap resampling test (Noreen (1989), Yeh (2000)) to the output of the classifier, which has also been used earlier in the framework of the CoNLL shared task on NER (Tjong Kim Sang and De Meulder 2003). This is done by randomly drawing feature vectors with replacement (bootstrap samples) from the classifier outputs. We repeated this step 1000 times. On the basis of these 1000 bootstrap results, we calculated the average F-score, the standard error and the upper and lower bound of the center 90% distribution. Since we do not know if the performance of our system is distributed according to a normal distribution, the significance boundaries are determined in such a way that for 5% of the samples the F-score was equal or below the lower significance boundary and that for 5% of the samples the F-score was equal or above the upper significance boundary. A score X is considered to be significantly different from a score Y if score Y is not within the center 90% of the distribution of X.

The results confirm that genetic algorithms can be successfully applied to the task of finding a classifier ensemble that outperforms the best individual classifier. However, the performance gains measured in our experiments are not as large as the ones reported in Ekbal and Saha (2010). One possible explanation for this is that the base classifiers used in their experiments were not as strong as the ones used in our experimental setup, leaving a bigger margin for improvement.

This raises doubts about whether ensemble classification can lead to better classification performance than a highly optimized individual classifier. In a brief experiment, the feature set of the best-performing CRF classifier was adapted to include the features of the second-best classifier it did not already have, namely prefixes and suffixes of length 3. This classifier achieved an F-score of 84.91 on the dataset, thus outperforming both the best individual classifier and the best ensemble classifier by 0.47 and 1.14 percentage points, respectively. These preliminary findings indicate that even larger performance gains might be achieved if structural feature selection and parameter optimization would be applied, as in the work described by Daelemans et al. (2003).

6 Conclusions and future work

This article has focused on the use of 3 different classification frameworks to construct a classifier ensemble for Dutch named entity recognition. The selection of the classifiers from a pool of 32 was done using a genetic algorithm. The re-

sults confirm that genetic algorithms can successfully be applied to the task of finding a classifier ensemble that outperforms the best individual classifier. The experiments also showed that combining different classification frameworks in an ensemble seems to benefit generalization performance. In future work, a detailed error analysis to determine the strengths and weaknesses of each classifier should be made. Another interesting alley for future work is the use of the output probabilities for every class label when combining the classifiers' votes, instead of using only the most probable class output by each classifier.

The performance gain of the ensemble system over the best individual classifier is statistically significant. However, it is not very large and comes at a high computational cost. In future work, we therefore intend to compare the use of genetic algorithms for ensemble selection to using them for the task of selecting features and optimizing parameters for one single classifier.

References

Bogers, T. (2004), *Dutch named entity recognition: Optimizing features, algorithms, and output*, Master's thesis, Universiteit van Tilburg.

Chinchor, N. (1998), Overview of MUC-7, *Proceedings of the 7th Message Understanding Conference*.

Chinchor, N. and P. Robinson (1997), MUC-7 named entity task definition, *Proceedings of the 7th Conference on Message Understanding*.

Cucerzan, S. (2007), Large-scale named entity disambiguation based on Wikipedia data, *Proceedings of EMNLP-CoNLL*, pp. 708–716.

Daelemans, W. and A. van den Bosch (2005), *Memory-based Language Processing*, Cambridge University Press.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch (2009), TiMBL: Tilburg Memory Based Learner, version 6.2, reference guide, *Technical Report 09-01*, ILK Research Group.

Daelemans, W., V. Hoste, F. De Meulder, and B. Naudts (2003), Combined optimization of feature selection and algorithm parameters in machine learning of language, *Machine Learning* pp. 84–95.

De Meulder, F. and W. Daelemans (2003), Memory-based named entity recognition using unannotated data, *Proceedings of the 7th Conference on Natural Language Learning*.

Desmet, B. and V. Hoste (2010), Towards a balanced named entity corpus for Dutch, *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Ekbal, A. and S. Saha (2010), Maximum entropy classifier ensembling using genetic algorithm for NER in Bengali, *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Isozaki, H. and H. Kazawa (2002), Efficient support vector classifiers for named entity recognition, *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*, Taipei, Taiwan.

Kudo, T. and Y. Matsumoto (2003), Fast methods for kernel-based text analysis, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 24–31.

Lafferty, J., A. McCallum, and F. Pereira (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *Machine Learning International Workshop*.

LDC (2008), *ACE (Automatic Content Extraction) English Annotation Guidelines for Entities Version 6.6*, Linguistic Data Consortium, Philadelphia, USA. http://projects.ldc.upenn.edu/ace/.

Noreen, E.W. (1989), *Computer Intensive Methods for Testing Hypothesis: An Introduction*, John Wiley & Sons, New York.

Oostdijk, N., M. Reynaert, P. Monachesi, G. Van Noord, R. Ordelman, I. Schuurman, and V. Vandeghinste (2008), From D-Coi to SoNaR: A reference corpus for Dutch, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Schuurman, I., V. Hoste, and P. Monachesi (2009), Cultivating trees: Adding several semantic layers to the Lassy treebank in SoNaR, *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, Groningen, The Netherlands.

Tjong Kim Sang, E.F. (2002a), Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition, *Proceedings of the 6th Conference on Natural Language Learning*, Taipei, Taiwan, pp. 155–158.

Tjong Kim Sang, E.F. (2002b), Memory-based shallow parsing, *Journal of Machine Learning Research* **2**, pp. 559–594.

Tjong Kim Sang, E.F. and F. De Meulder (2003), Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition, *Proceedings of the 7th Conference on Natural Language Learning*, Edmonton, Canada, pp. 142–147.

Vapnik, V. and C. Cortes (1995), Support vector networks, *Machine Learning* **20**, pp. 273–297.

Wang, H., T. Zhao, H. Tan, and S. Zhang (2008), Biomedical named entity recognition based on classifiers ensemble, *International Journal of Computer Science and Applications* **5** (2), pp. 1–11.

Whitley, D. (1994), A genetic algorithm tutorial, *Statistics and Computing* **4**, pp. 65–85.

Yeh, A. (2000), More accurate tests for the statistical significance of result differences, *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, Saarbrucken, Germany, pp. 947–953.

Extending Memory-Based Machine Translation to Phrases

Maarten van Gompel, Antal van den Bosch, and Peter Berck

Tilburg centre for Cognition and Communication
Tilburg University

Abstract

We present a phrase-based extension to memory-based machine translation. This form of example-based machine translation employs lazy-learning classifiers to translate fragments of the source sentence to fragments of the target sentence. Source-side fragments consist of variable-length phrases in a local context of neighboring words, translated by the classifier to a target-language phrase. We compare three methods of phrase extraction, and present a new decoder that reassembles the translated fragments into one final translation. Results show that one of the proposed phrase-extraction methods—the one used in Moses—leads to a translation system that outperforms context-sensitive word-based approaches. The differences, however, are small, arguably because the word-based approaches already capture phrasal context implicitly due to their source-side and target-side context sensitivity.

1 Introduction

Memory-based machine translation (van den Bosch and Berck 2009, van den Bosch et al. 2007, Canisius and van den Bosch 2009) (MBMT for short) is a variant of example-based machine translation. A key characteristic of MBMT is the use of memory-based classifiers (Daelemans et al. 1997, Daelemans et al. 2007) for the translation step. Memory-based classifiers do not only look up stored translation fragment pairs, but are also able to generate translations when the input does not offer an exact match in memory. A parallel corpus serves as the training material. All sentences in this parallel corpus are tokenised and paired up with their counterparts, and between the words of each sentence pair, an alignment is computed. This alignment serves as the basis from which small source-language fragments in their context can be extracted that are subsequently passed to a classifier for training. Whereas prior research in MBMT composed these fragments from single words in context (van den Bosch and Berck 2009, Canisius and van den Bosch 2009), the approach proposed in this study takes a phrase–one or more words—as the focal element of each fragment.

We thus start from a mapping of fragments in the source language to fragments in the target language extracted from a training corpus. Subsequently, memory-based learning is applied to convert these paired fragments into a memory-based classifier (Daelemans et al. 1997). This classifier can then be used to translate new sentences. Given a sentence to translate, we segment it into various phrase-based fragments; for each fragment, a distribution of possible output fragment translations is predicted by the memory-based classifier. As a final step, all translations

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

of these fragments are recombined by a new decoder that searches for a globally optimal translation of the given sentence.

The study builds upon previous research on MBMT (van den Bosch and Berck 2009, van den Bosch et al. 2007, Canisius and van den Bosch 2009). The question addressed here is whether a phrase-based approach improves MBMT. An extension to phrases introduces non-trivial issues; one is how to detect phrases in a parallel training corpus. In the study described in this paper, three methods of phrase extraction are tested and compared. Second, there is the issue of choosing a representation of variable-length phrases in the fixed-length feature vector representation assumed by memory-based classification.

In Section 2 we present our approaches to phrase-based MBMT in detail. In Section 3 the results of a comparative series of experiments are presented and discussed. We formulate our conclusions and starting points for future research in Section 4. The system presented in this paper is available as open source software from `http://ilk.uvt.nl/mbmt/pbmbmt`.

2 Phrase-based memory-based machine translation

An MBMT system divides into a training subsystem, producing a translation model, and a translation subsystem. Figure 1 illustrates the setup of the system. A parallel corpus is used for phrase extraction and example generation, i.e. the generation of translations of source fragments to target fragments. These fragments, with as their main constituent an aligned pair of phrases, are stored into a compressed tree structure during the training phase, this in order to facilitate fast retrieval, but as a side effect memory needs are minimized as well. In testing, unseen source-language sentences in a test corpus are also divided into fragments, which the memory-based classifier maps onto a distribution of target-language fragment translations. A decoder then reassembles all translated fragments together into one sentence, searching through and choosing among alternative solutions.

2.1 Example generation

We assume a word alignment between all sentence pairs in the parallel corpus. Figure 2 (left) illustrates such a word-aligned sentence pair, serving as an example throughout this section. On the basis of this, we create example fragment translations that serve as training examples. On the input side, an example consists of a feature vector representing a source-language fragment; on the output side, the example is labeled with a class, representing a fragment of the target sentence aligning to the source fragment. In prior research (van den Bosch and Berck 2009, Canisius and van den Bosch 2009), the feature vector consisted of one focus word, one context word to the left, and one context word to the right; the class was composed of the target-language word aligned to the focus word, and again one context word to the left, and one to the right. Suppose we translate French to English and look at the word est in Figure 2 (left), then the feature vector would be (inconnu, est, condamné), and the class would be (man,is,wrongly). Note that

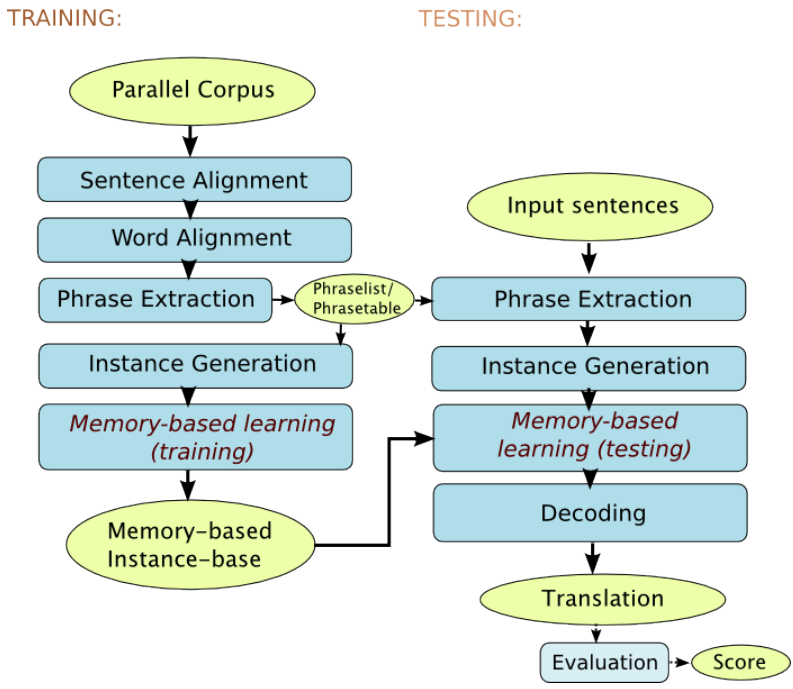


Figure 1: Setup of the phrase-based memory-based machine translation system. The left-hand side corresponds to the training phase; the right-hand side corresponds to the testing phase. Rounded nodes denote data, and square nodes denote processes that manipulate the data.

the class is considered by the classifier as an atomic symbol, but it is decomposed later into its constituents by the decoder. By moving a sliding window over the source sentence, fragments can be generated for all words save for zero-fertility words.

The phrase-based approach we present here is similar. Examples are composed as follows: The feature vector consists of a phrase from the source sentence, with one context word on the left side, and one context word on the right side. The class consists of the target-language phrase that aligns to the source-language phrase, and can optionally also take left and right context words. However, in our study, as detailed below, we found that taking no target-side context produced markedly better results. In representing the source-language side of the example as a feature vector, the variable-width focus phrase can be coded into multiple features. Since phrases can be of arbitrary length and the classifier expects a feature vector of fixed size, this poses a problem. Section 2.3 addresses this issue further.

Suppose that the alignment links the source-language phrase *"l'homme inconnu"* to the target-language phrase *"the unknown man"* in the target language.

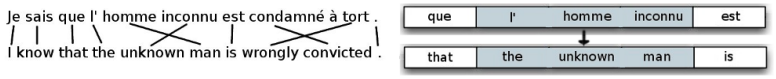


Figure 2: *Left:* A word alignment between a French and English sentence, *Right:* A phrase-based training example in context

Figure 2 (right) illustrates the phrase-based training example generated for this aligned pair of phrases.

2.2 Phrase extraction

A first task in this study is how to determine phrases in the source- and target-language sentences in the parallel corpus available for training. One solution is to employ the same type of method as used in phrase-based statistical machine translation, making use of a phrase-translation table (Koehn 2004). Such a table lists aligned phrases in both source and target language, assigning conditional probabilities to each. These aligned phrase pairs are computed statistically over the entire parallel corpus by taking the intersection of source-to-target and target-to-source word alignments (Koehn 2004), and this can subsequently be extended by using an algorithm that incrementally adds certain points from the union of the two alignments (Och and Ney 2003). We use the implementation in Moses (Koehn et al. 2007) to this end.

In addition to this first method, we include two other approaches to phrase extraction for comparison. The second method of phrase extraction, henceforth named the *phrase-list approach*, is a straightforward method that extracts frequent n-grams only from the source-language side of the training corpus, and stores this in what we call a *phrase list*. The approach needs a frequency threshold above which an n-gram is included in the phrase list. After exploratory experimentation on test material this threshold was set at 25, but more extensive optimisation can be conducted. Unlike in the phrase-table method, the aligned counterpart of a source-side phrase is computed on the fly. Each source sentence is matched against the phrase list, and whenever a phrase is found, we follow the word-alignments from the phrase and assume that the sequence of words it points to is the aligned target phrase, possibly with intervening fertility words.

Using phrases from either a phrase-translation table or phrase-list, we can never expect to obtain full coverage of test sentences. To decrease problems of low coverage and data sparsity, we defined a phrase to consist of *one or more* words. In addition to phrase extraction, we always generate word-based fragments using the same word-oriented approach as used in prior research. This makes the phrase-based approach an extension of the word-based approach. Given the same parallel corpus and input sentences, the training and test examples in phrase-based MBMT are a superset of those in word-based MBMT.

Due to the phrase-based character of our approach, a word in the source sen-

tence can be part of the focus of a feature vector multiple times. A single word always generates its word-based example, but there may also be one or more extracted phrases that the word is a part of. This occurs in both training and test examples. The latter has an important side effect that has an impact on the decoding process we describe below: if there are multiple examples covering the same words, then there will be multiple possible fragmentations of the input sentence (see also Figure 4).

The third phrase-extraction method is *marker-based chunking*, which segments a sentence into non-overlapping chunks, splitting whenever so-called marker words occur. Marker words are typically defined as closed-class function words, and overlap significantly with the most frequent words in most corpora. Marker-based chunks are generated whenever a new marker word occurs at the beginning of the sentence or after a non-marker (or content) word. Thus, each chunk contains at least one non-marker word. When generating training instances, both the source sentence and the target sentence are chunked independently. Figure 3 shows an example of marker-based chunking, in which arrows point at the markers, each at the head of a chunk. The idea behind marker-based chunking is rooted in the Marker Hypothesis (Green 1979), an idea from psycholinguistics that posits that all languages are marked for surface syntax by a specific closed set of lexemes or morphemes (van den Bosch et al. 2007). Marker-based chunking is a phrase extraction strategy that differs from the previous two in the sense that it does not use word n-gram statistics. It has already been employed in a previous study of MBMT (van den Bosch et al. 2007), inspired in turn by its earlier application in EBMT (Gough and Way 2004, Way and Gough 2005).

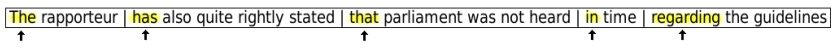


Figure 3: Example of marker-based chunking.

The next step is to align the marker-based chunks in the source and target sentences, on the basis of the word alignments already at our disposal. This procedure, described in (van den Bosch et al. 2007), aims to find the target chunk that has the highest probability of being aligned to the source chunk. We effectively estimate $P(C_t|C_s)$ for each source chunk—where C_s is a chunk in the source sentence and C_t a chunk in the target sentence—and align the source chunk with the most probable target chunk. For accurate results the alignment needs to be performed in the other direction as well, estimating $P(C_s|C_t)$ and aligning each target chunk with the most probable source chunk. The intersection of both alignments is then taken as the resulting most likely chunk alignment.

2.3 Representing phrase-based examples

If we encode the focus phrase of the feature vector in terms of its words and their position in the phrase, we end up with feature vectors of different sizes. However, the memory-based classifier demands a fixed number of features in order to compute its similarity function. There are at least three ways to resolve this problem. First, we can consider the entire phrase as one atomic feature. Second, we can reserve a fixed number n of features, and fill those with position-specific words (such as the final word, the prefinal word, etc.), assigning dummy values to unused feature slots. Third, we can assign separate classifiers to different phrase lengths, assigning examples with a particular phrase-length to a separate classifier trained only on examples of this length. In this setup a master process assigns examples to different classifiers and reassembles their output again for the decoder. All three methods of representation have been tried in our research.

2.4 Decoding

Due to the overlapping nature of extractable phrases, and the fact that we may end up with multiple examples covering the same words in the source sentence, we can speak of various possible *fragmentations* of the source sentence S. We define a fragmentation to be a chain in which the fragments are non-overlapping; each fragment covers a certain range of consecutive words of arbitrary length n in the source sentence S, where $1 \leq n \leq |S|$. In addition each fragment is associated with a left context and right context of a length predetermined during example generation. Figure 4 shows three example fragmentations.

Each test example is mapped by the classifier to a distribution of classes, which are the various target-side translations for the fragment with an associated probability score. This probability score is derived from the class distributions produced by the memory-based classifiers by normalizing the class votes. From the perspective of the decoder, the target-side translations with probability scores are referred to as *hypothesis fragments*. Thus, each source-side fragment will be associated with a collection of one or more hypothesis fragments. Figure 4 illustrates the relation between the fragmentation of a sample sentence, the source-side fragments that are extracted from it, and the target-side hypothesis fragments generated from the source-side fragments.

Having gathered all matching fragments for a given source sentence, the task is to search for "good" fragmentations, leading to the most likely translation. The number of fragmentations tends to increase exponentially in the length of the source sentence. Although we do classify all of them with the memory-based classifier, it is infeasible to subsequently search in the space of all possible rearrangements in complete output sentences. Therefore, a local beam search is used to select a number of good fragmentations. The beam size is rather arbitrarily fixed at 20, so at most 20 fragmentations will be returned. The beam search incrementally adds fragments, maximising a score function that sums the normalized scores of the most likely hypothesis fragments predicted for each source-side fragment

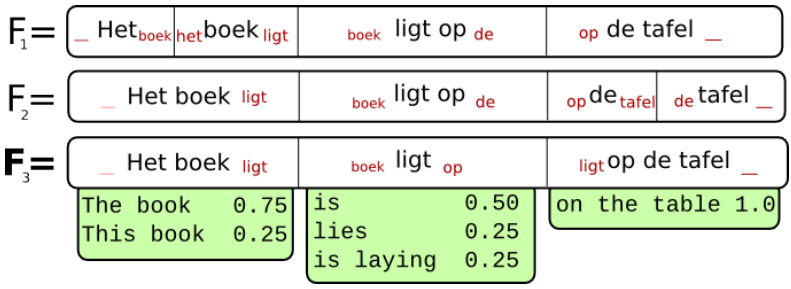


Figure 4: Fragmentations of the sample Dutch input sentence *"Het boek ligt op de tafel"* (The book is on the table). The third fragmentation is expanded to list the target-side hypothesis fragments associated with each of the three source-side fragments the fragmentation is composed of. Context information is printed in small text.

in the current fragmentation. Only fragmentations covering the full sentence are considered as possible solutions.

The fragmentation search described above returns a maximum of 20 fragmentations. For each of these fragmentations of the source sentence, the sentence-global decoding procedure is started. This procedure employs another local beam search to search among alternative translation hypotheses for the given fragmentation. There are thus two search processes going on; one for good fragmentations, and one to search translation hypotheses for each good fragmentation. This makes the phrase-based approach computationally more expensive compared to the word-based approach, as the latter only has one fragmentation of the source sentence.

The actual decoding procedure starts by generating an initial hypothesis: a translation hypothesis in which we simply select for each fragment in the fragmentation the hypothesis fragment with the highest translation probability. We order the hypothesis fragments for the initial hypothesis in the order we find the fragments in the source-sentence fragmentation. The initial hypothesis in Figure 4 thus is *"The book is on the table"*. In this example, the initial hypothesis already happens to generate the best translation, but in most cases there is more searching to do. A hypothesis can be modified in two main ways: (1) the order in which the hypothesis fragments are assembled can be changed, and (2) the choice of hypothesis fragments can be changed, i.e. other hypothesis fragments with an equal or lower translation probability could be tried. To this end, the decoder applies two operations to the initial hypothesis. Each yields new hypotheses, and the best, limited by a beam, are selected. To these hypotheses the operators are applied again. This procedure repeats itself until no better scoring hypotheses can be generated. The first operator is **substitution**. It generates new hypotheses in which a hypothesis fragment of a particular fragment is substituted by another hypothesis fragment from the list. This is done exhaustively within the list of hypothesis fragments (bounded by the first beam search). For each fragment, substitutions are

made using all hypothesis fragments that have not undergone a substitution operation in a previous decoding round. The second operator is the **swap** operation, in which we swap the location of two hypothesis fragments. This again is done in an exhaustive fashion such that all possible swaps are made. Each fragment swaps places with all neighboring fragments that are within a certain maximum swap distance, each swap yielding a new hypothesis. An extra parameter specifies the maximum range over which a swap can occur; we set this at two, but exploratory optimisation showed changing this value had little to no effect, the reason is still unknown and is a topic that will be addresses in future research.

In sum, the core decode process, implemented as a local beam search, is called with the initial hypothesis, and then computes all possible substitutions and all possible swaps, resulting in a high number of new hypotheses. This constitutes an informed search problem that seeks to maximise the score of the solution hypotheses. The k best hypotheses are selected, k being the width of the beam, with the constraint that they must be better than the current hypothesis. For each of these k hypotheses the procedure is again repeated, calculating all possible substitutions and swaps, until the point that no better hypotheses can be found. The algorithm can be formalised as in Algorithm 1:

Algorithm 1 Core-Decoder

Require: A set $X_{current}$ containing hypotheses, initially called with only the initial hypothesis as element, and a beam width k.

Ensure: The list containing the k best hypotheses found

$X_{next} \Leftarrow \emptyset$
for all $H_{current} \in X_{current}$ **do**
  for all $H_{next} \in Substitutions(H_{current}) \cup Swaps(H_{current})$ **do**
    if $Score(H_{next}) > Score(H_{current})$ **then**
      $X_{next} \Leftarrow X_{next} \cup \{H_{current}\})$
    end if
  end for
end for
if $X_{next} == \emptyset$ **then**
  return $X_{current}$
else
  $X_{next} \Leftarrow$ Best k hypotheses in X_{next}
  return $CoreDecoder(X_{next}, k)$
end if

The success of the decoding algorithm depends on the sentence-level score function it maximises. For each hypothesis, a score is computed that expresses the quality of the hypothesis. A good translation should preferably maximise both *fidelity* and *fluency*. Quantified estimators for these two components are present in the decoder developed for the present study. A quantification of fluency is provided by a trigram-based statistical language model with back-off smoothing and

Good-Turing smoothing on the target language (Stolcke 2002), while fidelity is expressed by the normalized pseudo-probabilities generated by the memory-based translation model. More precisely, the score function for a hypothesis H is made up of the product of the translation probability and the distortion score of the given hypothesis, as shown in Equation 3.1:

$$\text{TranslationModel}(H) = \text{ClassifierScore}(H) \cdot \text{DistScore}(H) \quad (3.1)$$

$$\text{ClassifierScore}(H) = \prod_{i=1}^{|H|} P(classification_i^{weight}) \quad (3.2)$$

$$\text{DistScore}(H) = \prod_{i=1}^{|H|} distortion\_constant^{distance(hfragment_i, hfragment_{i-1})} \quad (3.3)$$

The sentence-level score $\text{ClassifierScore}(H)$ (Equation 3.2) is the product of the translation probabilities of all selected hypothesis fragments that make up the hypothesis. Recall that these translation probabilities are derived from the classifier output, which predicted a distribution of hypotheses fragments as illustrated in Figure 4. The translation probabilities can be given an extra weight parameter by raising them to a certain power.

In addition, a distortion score $\text{DistScore}(H)$ (Equation 3.3) is computed by raising a distortion constant to the power of a measure of distance between two fragments in the original source sentence. The distortion score thus penalises the reordering of fragments; the greater the distance over which an hypothesis fragment is relocated, the lower the distortion score will be. The distortion constant itself is a value between 0 and 1, where the former disallows any reordering of the hypothesis fragments, and the latter does not impose any restrictions at all. This is an admittedly crude factor that may fit the language pair, but will tend to favor ungrammatical and undistorted target sequences over grammatical but reordered target sequences.

One straightforward manner in which to select the final translation would be to select the hypothesis that has the highest score. Yet, this foregoes the fact that among the solutions considered by the decoder, there may be hypotheses representing the same output sentence, even though they are composed of different hypothesis fragments. In the final solution search algorithm we therefore take the sum of the scores of all hypotheses that produce the same output sequence of words in the target language across all of the hypotheses considered.

The last remaining step is to select the target-language sentence for which the sum of the scores of the hypotheses that generated this sentence is maximal. This is the translation generated by the system.

3 Results

Experiments were performed on two parallel corpora, in which we focused only on Dutch to English translation. The first is OpenSubtitles (Tiedemann and Nygaard 2004) consisting of user-contributed subtitles for movies. We generated

a training set split consisting of $286,160$ sentence pairs. The second corpus is EMEA (Tiedemann and Nygaard 2004), a medical and largely formulaic text corpus from which we split a training set of $871,180$ sentence pairs. From both corpora we also extracted non-overlapping development and test sets of $1,000$ sentences each.

Several exploratory parameter optimisation experiments were carried out on development data in order to assess the effect of the decoder and some of the parameters. One outcome of these explorations is that omitting target-side context, as used in prior research where target-side fragments constituted trigrams of words (van den Bosch and Berck 2009, Canisius and van den Bosch 2009), greatly improves results. In an experiment on the OpenSubtitles corpus, the BLEU score on development data increased from 0.1211 with target-side context to 0.2184 when target-side context is removed and only target-side phrases are predicted. We attribute this effect mainly to the increase in sparsity of classes when adding context, adding to the sparsity of the phrases themselves.

The main question addressed in this study is whether a phrase-based approach to MBMT (PBMBMT) improves upon the previous word-based approaches (MBMT (van den Bosch and Berck 2009), CSIMT (Canisius and van den Bosch 2009)). In both these methods, the feature vector as well as the class consist of a trigram: one focus word, one left-context word, and one right-context word. CSIMT employs a decoder based on Constraint Satisfaction Inference, and searches through the space of possible target-level translations on the basis of a hill-climbing search (Germann 2003). As our decoder performs a similar search, we compare against this variant. For completeness, we also compare to the MBMT variant described in (van den Bosch and Berck 2009) that maps source trigrams to output trigrams, and uses a decoder that is purely based on target trigram overlap. This overlap-based decoder was introduced in (van den Bosch et al. 2007), where it was shown to outperform a marker-based variant of MBMT.

Other subquestions addressed in this study are: How do the three phrase-extraction methods perform and compare? What example format is best? Indications for answering these questions can be found in Table 3.1. Note that in this table we also list results produced by the PBMBMT decoder when run without using any phrase-extraction method (named wb-PBMBMT), making it operate on a word-based level similar to word-oriented CSIMT, with the notable difference that target-side context is excluded in all PBMBMT experiments. This word-based system offers a baseline for assessing the effectiveness of the phrase-extraction methods, and it can be compared to the word-based decoders reported in earlier work (van den Bosch and Berck 2009, Canisius and van den Bosch 2009).

With respect to the three phrase extraction methods, the Moses (Koehn et al. 2007) phrase-table method performs best overall. The other two methods, especially marker-based chunking, perform below the word-based PBMBMT baseline. This may be attributed to the fact that the phrase-translation table is computed using the two-way alignment statistics of the parallel corpus, whilst the other two methods only rely on source-side statistics, and the aligned counterparts of the phrases are sought in an ad-hoc and per-sentence fashion. The two predecessor

OpenSubtitles

Decoder	Extraction Method	Single/multi classifier	Translation performance metrics BLEU	NIST	METEOR	WER	PER
MBMT	-	-	0.1631	4.243	0.3835	68.39	61.33
CSIMT	-	-	0.2002	4.750	0.4431	68.42	55.18
wb-PBMBMT	-	-	0.2163	**5.136**	**0.4644**	55.23	**48.22**
PBMBMT	phrase table	single	**0.2300**	5.055	0.4623	54.47	49.18
PBMBMT	phrase table	multi	0.2256	5.004	0.4583	55.28	49.74
PBMBMT	phrase table	atomic	0.1142	3.026	0.3201	72.81	68.28
PBMBMT	phrase list	single	0.2190	4.980	0.4543	**54.09**	48.77
PBMBMT	phrase list	multi	0.2184	4.975	0.4529	**54.09**	48.79
PBMBMT	marker-based	single	0.1003	2.935	0.3057	76.79	71.16
PBMBMT	marker-based	multi	0.1394	3.360	0.3437	66.40	62.38

EMEA

Decoder	Extraction method	Single/multi classifier	Translation performance metrics BLEU	NIST	METEOR	WER	PER
MBMT	-	-	0.2533	5.115	0.4801	72.78	63.66
CSIMT	-	-	0.3013	5.938	0.5333	63.00	**50.85**
wb-PBMBMT	-	-	0.2715	5.600	0.5381	65.99	57.25
PBMBMT	phrase table	single	0.3075	6.011	**0.5455**	59.00	52.02
PBMBMT	phrase table	multi	**0.3078**	**6.019**	0.5449	**58.76**	51.63
PBMBMT	phrase list	single	0.2440	5.352	0.4946	62.74	56.67
PBMBMT	phrase list	multi	0.2440	5.378	0.4967	62.82	56.86
PBMBMT	marker-based	multi	0.2370	4.612	0.4513	74.37	66.78

Table 3.1: Main results on the **OpenSubtitles** and **EMEA** corpora, Dutch to English. Selected parameters were a distortion constant of 0.25, a translation weight of 3, and a maximum swap distance of 5, and a beam width of 1. Note however that in later experiments, slightly higher results have been reported with a beam width of five.

systems, MBMT (van den Bosch and Berck 2009) and CSIMT (Canisius and van den Bosch 2009), are both outperformed by the Moses phrase-table method, and as reported earlier in (Canisius and van den Bosch 2009), the CSIMT method in turn tends to outperform the MBMT method with the trigram overlap-based decoder on all metrics. On the EMEA corpus, CSIMT outperforms the word-based PBMBMT, and performs relatively close to PBMBMT with the Moses phrase-table approach.

We thus observe that the advantage of phrase-based MBMT compared to earlier word-oriented approaches, including the closest comparable system, the word-based PBMBMT baseline (wb-PBMBMT in the table) that restricts itself to words and uses the same decoder as PBMBMT, turns out to be limited. This is a surprising outcome. We may posit that sparsity plays a role here; phrases are by definition less prevalent than single words. The omission of context in classes (in contrast to CSIMT, which maps to trigrams of words) attempts to compensate for this to a certain extent. Another reason for the lack of a clear difference between word-based and phrase-based MBMT may be sought in the fact that even in word-oriented CSIMT there is already a significant but implicit presence of phrasal context. Essentially we are comparing *phrasal context inherent to the phrases themselves* in

PBMBMT, against *phrasal context implicit in the input and output word trigrams* in CSIMT. Often, and most clearly with phrases of three words, the two approaches are mapping about the same input to the same output. The limited gain of the phrase-based approach may stem from the added value of the fact that PBMBMT is not restricted to trigrams, and can vary between whatever is the strongest n-gram, including unigrams. We found that in an experiment on the OpenSubtitles corpus, using the phrase-table method and multi-classifier format, on average 78% of the hypothesis fragments of the predicted translations is in fact a unigram.

Concerning the example format, reserving space for a fixed number of position-specific features (i.e. words) in a single classifier versus distributing different phrase-lengths over multiple classifiers perform more or less on a par. A third format, in which we took a phrase to be an atomic entity, rather than splitting the word over different features, proves to be a poor method of representation, as it yields significantly lower scores (see the entry marked "atomic" in Table 3.1).

In addition to these comparisons, we compared the phrase-based memory-based machine translation approach to state-of-the-art machine translation approaches. Table 3.2 compares PBMBMT with three other systems. The first two, Moses (Koehn et al. 2007) and Google Translate[1], are phrase-based statistical machine translation systems, while Systran is a largely rule-based system. Phrase-based memory-based translation does not approach the performance of the statistical systems. On the other hand, PBMBMT outperforms Systran on all metrics.

No	Corpus	System	BLEU	NIST	METEOR	WER	PER
1	OpenSub	Moses	**0.3289**	**5.903**	**0.5408**	53.29	46.96
2	OpenSub	Google	0.3056	5.790	0.5224	**50.1**	**45.08**
3	OpenSub	PBMBMT	0.2300	5.055	0.4623	54.47	49.18
4	OpenSub	Systran	0.1749	4.583	0.4500	60.77	54.61
1	EMEA	Moses	**0.4701**	**7.059**	**0.6501**	**46.55**	**39.36**
2	EMEA	Google	0.3918	6.377	0.5830	57.57	50.44
3	EMEA	PBMBMT	0.3075	6.011	0.5455	59.00	52.02
4	EMEA	Systran	0.2895	5.472	0.5366	63.24	55.14

Table 3.2: A comparison with state-of-the-art systems

We do not yet have a clear insight into why exactly PBMBMT underperforms in relation to state-of-the-art Phrase-based Statical Machine Translation systems such as as Moses. One cause may be the extensive usage of minimal error training (MERT) on development material, for hyperparameter tuning in Moses, which we have not explored.

[1] `http://translate.google.com`

4 Conclusions and future research

The study described in this paper has demonstrated how memory-based machine translation can be extended from translating fixed-length word trigrams to translating phrases of arbitrary length. We compared three methods of phrase extraction, of which the Moses phrase-translation table approach emerges as the best solution, in fact the only solution scoring above baseline.

Prior research in MBMT such as the recent CSIMT approach (Canisius and van den Bosch 2009) relied partly on target-side context, making use of the overlap between predicted target-side hypothesis fragments (word trigrams) in decoding. The current study shows that ignoring target-side context produces significantly better results in a phrase-based approach, and even performs well in a word-based mode. This can be credited to the decrease in sparsity in the output class space. Moreover, removing this context can be justified by the fact that context becomes less relevant in phrase-based approaches, as target-side phrases capture enough internal context themselves.

Nevertheless, the impact of phrases in comparison to word-based MBMT has been shown to be limited. A potential explanation for this limited effect is that earlier word-based MBMT approaches can be seen as implicitly phrase-based already. The approach followed in (van den Bosch and Berck 2009, Canisius and van den Bosch 2009) maps trigrams of source-side words to trigrams of target-side words, implicitly capturing all phrases up to length three. In this perspective, our current approach changes this only slightly by turning the source-side trigrams into variable-width phrases surrounded by their left and right neighboring words, and predicting variable-width target-side phrases, including single words. In one of our experiments, we found that 78% of fragments in the predicted translation, consists of such single words.

With respect to decoding, we observed in preliminary experiments that starting with an initial hypothesis that follows the order of the source-side fragments as a starting point is a better heuristic than starting with an empty hypothesis and incrementally adding hypothesis fragments. Moreover, subsequently applying substitution and swap operations appears a viable strategy for improving upon this initial hypothesis, even though the resulting gain is modest. Phrase-based decoding differs from word-based decoding in that it needs to take into consideration alternative fragmentations of the input sentence. A strategy has been employed that starts the decoding process with a beam-constrained number of best fragmentations (where goodness is estimated from the pseudo-probabilities generated by the memory-based classifier), and in the end combines the results to select the highest scoring candidate. There are thus essentially two inter-dependent search problems to be solved. Despite the fact that this increases the complexity of decoding, the strategy of applying substitutions and swaps in a hill-climbing search until convergence, results in performing the decoding task in a limited time-span.

Besides hyperparameter optimisation of both the memory-based classifier and the decoder through minimal error training, other options for future research are testing the system on different, more distantly related, language pairs, and the in-

clusion of richer (e.g. linguistic) features such as part-of-speech tags and lemmas. We believe that considerable improvement can be obtained by improving the decoder. In future work it could be extended with more operations, such as a delete operation powered by a null model; moreover, an alternative should be sought for its current crude distortion factor. The findings with regard to omission of target-side context could be tested and incorporated into the strategy proposed in CSIMT (Canisius and van den Bosch 2009).

Acknowledgement

This research was funded by NWO, the Netherlands Organisation for Scientific Research, as part of the Vici project "Implicit Linguistics".

References

Canisius, S. and A. van den Bosch (2009), A constraint satisfaction approach to machine translation, *Proceedings of the 13th Annual Conference of the European Association for Machine Translation (EAMT-2009)*, pp. 182–189.

Daelemans, W., A. van den Bosch, and T. Weijters (1997), Igtree: Using trees for compression and classification in lazy learning algorithms., *Artificial Intelligence Review* **11** (1-5), pp. 407–423. http://dblp.uni-trier.de/db/journals/air/air11.html#DaelemansBW97.

Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch (2007), Timbl: Tilburg memory based learner, version 6.1, reference guide, *Technical Report ILK-07-07*, Tilburg University, Tilburg, The Netherlands.

Germann, U. (2003), Greedy decoding for statistical machine translation in almost linear time, *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 1–8.

Gough, N. and A. Way (2004), Robust large-scale EBMT with marker-based segmentation, *Proceedings of the Tenth Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2004)*, Baltimore, Maryland, pp. 95–104.

Green, T. (1979), The necessity of syntax markers. two experiments with artificial languages, *Journal of Verbal Learning and Behavior* **18**, pp. 481–496.

Koehn, P. (2004), Pharaoh: A beam search decoder for phrase-based statistical machine translation models., *in* Frederking, R.E. and K. Taylor, editors, *Proceedings of the American Machine Translation Association*, Vol. 3265 of *Lecture Notes in Computer Science*, Springer, pp. 115–124.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst (2007), Moses: Open source toolkit for statistical machine translation., *ACL*, The Association for Computer Linguistics.

Och, F.J. and H. Ney (2003), A systematic comparison of various statistical alignment models., *Computational Linguistics* **29** (1), pp. 19–51. http://dblp.uni-trier.de/db/journals/coling/coling29.html#OchN03.

Stolcke, A. (2002), Srilm – an extensible language modeling toolkit. http://citeseer.ist.psu.edu/stolcke02srilm.html.

Tiedemann, J. and L. Nygaard (2004), The OPUS corpus - parallel and free, *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC04)*, pp. 26–28.

van den Bosch, A. and P. Berck (2009), Memory-based machine translation and language modeling, *The Prague Bulletin of Mathematical Linguistics* **91**, pp. 17–26.

van den Bosch, A., N. Stroppa, and A. Way (2007), A memory-based classification approach to marker-based EBMT, *Proceedings of the METIS-II Workshop on New Approaches to Machine Translation*, pp. 63–72.

Way, A. and N. Gough (2005), Comparing example-based and statistical machine translation, *Natural Language Engineering* **11** (3), pp. 295–309.

Finding Constraints for Semantic Relations via Clustering

Sophia Katrenko and Pieter Adriaans

University of Amsteram, the Netherlands

Abstract

Automatic recognition of semantic relations constitutes an important part of information extraction. Many existing information extraction systems rely on syntactic information found in a sentence to accomplish this task. In this paper, we look into relation arguments and claim that some semantic relations can be described by constraints imposed on them. This information would provide more insight on the nature of semantic relations and could be further combined with the evidence found in a sentence to arrive at actual extractions.

1 Introduction

Semantic relations have been an object of study for a long time and across different disciplines (Khoo and Na 2006). Within computational linguistics, the main focus has been on identifying relations automatically (McDonald 2005) and further use of the extracted relations in various applications to improve their performance (van der Plas 2008).

When talking about relations, we distinguish between their extension and intension. For the n-ary relation its extension is determined by the set of ordered entities (of size n) that satisfy it. For example, for the relation PART - WHOLE this set would have a member <professor, faculty> but not <faculty, professor>. The intension of a relation is defined by what it means (what does it mean that x is part of y?). More generally, one can represent the extension of an n-ary relation by a subset the Cartesian product of the sets $S_1, \ldots, S_n$ where each set corresponds to the particular argument of the relation. If two entities x and y are in the binary relation $\mathcal{R}$, we write $x\mathcal{R}y$ or $\mathcal{R}(x, y)$. A set of ordered entities that satisfy $\mathcal{R}$ (for instance, $< x, y >$) is referred to as instances of $\mathcal{R}$ or its mentions. Note that the same extensions do not necessarily mean the same intension. Two relations such as STUDY-IN and LIVE-IN may have the same extension but their intension is different.

Past research has led to studying similarity effects that can be imposed on relations. For instance, Chaffin and Herrmann (2001) distinguish between *item similarity* and *relation similarity* whereby the first measure, in contrast to the second, is constant and does not depend on the relation at hand. More precisely, if one is given a word pair $< x, y >$, item similarity is defined between arguments x and y, while relation similarity measures how close $< x, y >$ is to the target relation. Item similarity plays a significant role only for some relations like SYNONYMY or ANTONYMY where such similarity effects are clearly involved. Chaffin and

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

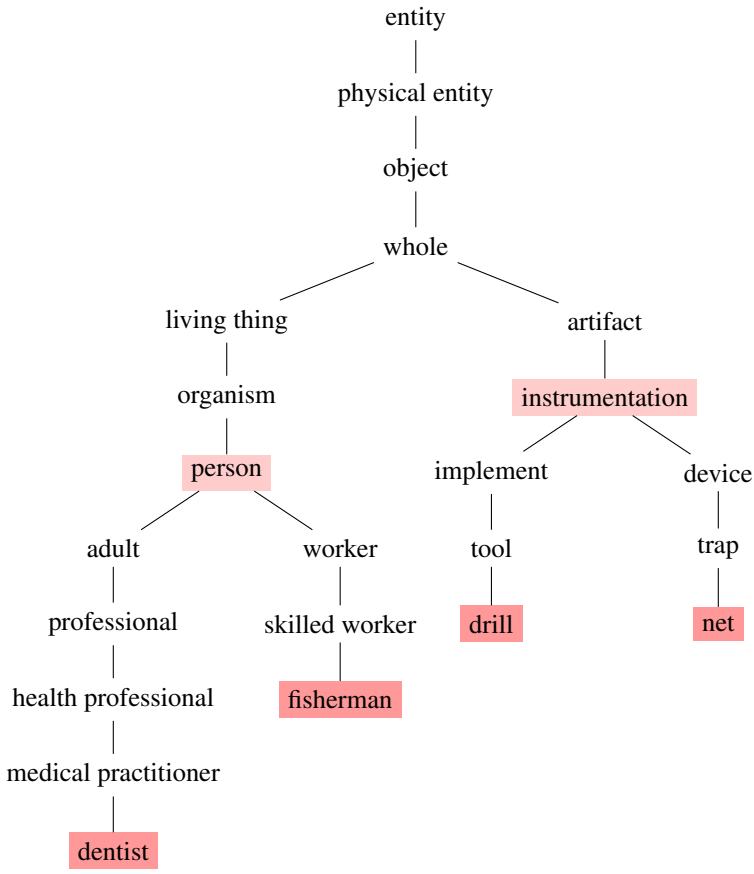


Figure 1: Part of the WordNet hierarchy.

Herrmann (2001) have studied yet another relation, PART-WHOLE, and showed that even though a part is not necessarily similar to a whole, there are still effects similar to those reported on other semantic relations. The authors concluded that relation similarity facilitates relation recognition and impedes negative response. As expected, item similarity did not contribute much to the recognition task and when held constant, relation similarity still affected the resulting performance.

While item similarity may not be applicable to all types of semantic relations, one can assume that argument fillers can be grouped according to their similarity. Consider, for example, the following two sentences.

(4.1) I saw a *fisherman* cleaning his *net*.

(4.2) One of the instruments a *dentist* uses often is a *drill*.

Here, we have two examples of the INSTRUMENT - AGENCY relation, <fisherman, net> and <dentist, drill>. If we look at the fragment of the WordNet hierarchy depicted in Figure 1, we could arrive at the generalization of these examples, such as <person, instrumentation>. It can serve as a semantic constraint for INSTRUMENT - AGENCY, because knowing that there is a person and an instrument we may conclude that it is likely for a person to use an instrument.

In this paper, we propose a method to derive constraints for semantic relations. In other words,

> *Given positive examples of a binary relation $\mathcal{R}(x, y)$ and a taxonomy $\mathcal{T}$, our goal is to find all possible pairs $< \mathcal{G}_x, \mathcal{G}_y >$ such that $\mathcal{G}_x$ is a semantic type of x and $\mathcal{G}_y$ is a semantic type of y, respectively.*

The paper is structured as follows. We introduce the method in Section 2 and proceed with the description of seven generic semantic relations (Section 3.1) and the experimental results (Section 3.2). In Section 4, we discuss our findings in more detail.

2 Methods

To find constraints for semantic relations, we describe a method which is based on positive examples only and does not make use of negative examples. To determine generalized semantic types of relation arguments, one has to be able to form clusters based on the existing information. Such clusters can be formed using semantic measures defined over WordNet (Fellbaum 1998). In particular, given argument x of $\mathcal{R}(x, y)$ and n corresponding synsets collected from the training data set ((x_1, y_1),. . .,(x_n, y_n)), we create a matrix S of size $n \times n$ by comparing each pair of synsets (s_i, s_j), $i = 1 \ldots n$. Each element of this matrix is equal to a similarity score of (s_i, s_j), with the diagonal elements equal to 1 (we assume that a similarity measure returns values from 0 to 1 with 1 being an identity score).

To form the matrices that can be used for clustering, we have to compare each pair of synsets for x and do the same for argument y. There exists a range of similarity measures that allow to compare a pair of synsets (Budanitsky and Hirst 2006). For our purpose, we selected wup measure which uses a notion of path length between two synsets. Given two synsets s_1 and s_2 connected by a path of length $len(s_1, s_2)$ and their least common subsumer $LCS(s_1, s_2)$, the wup score is calculated as follows (Palmer and Wu 1995):

$$wup(s_1, s_2) = \frac{2 * depth(LCS(s_1, s_2))}{len(s_1, s_2) + 2 * depth(LCS(s_1, s_2))} \tag{4.3}$$

Once a matrix S is obtained, we perform clustering. Ideally, the resulting clusters should reflect the semantic types of a given argument. However, to be able to use such clusters in future, we have to label them. This can be accomplished by using the least common subsumer. For each cluster C, $c_i \in C, i = 1, \ldots, k$,

$LCS(c_1, \ldots, c_k)$ is computed. This LCS corresponds to the semantic type $\mathcal{G}_x$ we are looking for.

Definition 1 (Least Common Subsumer)**.** *Given two nodes $\mathcal{N}_1$ and $\mathcal{N}_2$ in a taxonomy $\mathcal{T}$, their least common subsumer LCS is an ancestor of both $\mathcal{N}_1$ and $\mathcal{N}_2$ such that there is no node $\mathcal{C}$ which is ancestor of $\mathcal{N}_1$ and $\mathcal{N}_2$ and a child of LCS.*

Recall that such generalization is done per argument and we need to find pairs of clusters that would correspond to $R(x, y)$. Let l be a number of clusters for x and m be a number of clusters for y. To find cluster pairs, we introduce a strength coefficient between any pair of clusters as follows. For each cluster $C_i, i = 1, \ldots, l$ for the argument x, and for each cluster $C_j, j = 1, \ldots, m$ for the argument y, the strength coefficient $s(C_i, C_j)$ is calculated in the following way:

$$s(C_i, C_j) = \frac{\#links(C_i, C_j)}{min(|C_i|, |C_j|)} \tag{4.4}$$

In Equation 4.4, $\#links(C_i, C_j)$ stands for the number of links between elements of C_i and C_j. It is easy to see that if a mapping from C_i to C_j is injective, the strength coefficient can be at most 1 (all elements of one cluster are connected with some/all elements of the other) and at least 0 (there are no elements in both clusters that are connected with each other). If a mapping is surjective, then s can be larger than 1.

Using clustering to detect semantic types of arguments poses a problem of defining a number of resulting clusters. If the number of clusters is high, we expect to obtain specific generalizations and high precision/low recall. Reducing a number of clusters will most likely lead to less precise generalizations but higher recall.

There exist many clustering methods and it is clear that a choice of a clustering method may affect the results. However, we abstract away from a clustering approach by choosing a simple agglomerative method (Zhao and Karypis 2002).

The shortest path Provided that semantic constraints are identified with high recall, one may combine this outcome with syntactic evidence. One way of doing this is to consider all positive predictions by both syntax-based method and semantic constraints as positive in the final model, while the rest should be labeled as negative examples. To obtain predictions based on syntactic information, we use the shortest path kernel, which represents a kernel-based approach for relation extraction and explores information found in dependency trees (Bunescu and Mooney 2005). As input for this method serve dependency paths connecting two relation arguments. The more similar these paths are, the more likely two relation examples belong to the same category. Given the data sparseness problem, the authors generalize over existing paths by adding information sources, such as part of speech (PoS) categories or named entity types.

The shortest path between relation arguments is extracted and a kernel between two sequences (paths) $\mathbf{x} = \{x_1, \ldots, x_n\}$ and $\mathbf{x}' = \{x'_1, \ldots, x'_m\}$ is computed as follows:

$$k_B(\mathbf{x}, \mathbf{x}') = \begin{cases} 0 & m \neq n \\ \prod_{i=1}^{n} f(x_i, x'_i) & m = n \end{cases} \tag{4.5}$$

In Equation 4.5, $f(x_i, x'_i)$ is the number of features shared by x_i and x'_i. The features which are used as input are the following: word (e.g., $protesters$), part of speech tag (e.g., NNS), generalized part of speech tag (e.g., $Noun$), and entity type (e.g., $PERSON$) if applicable. In addition, a direction feature ($\rightarrow$ or $\leftarrow$) is employed.

3 Evaluation

In this section, we will describe seven semantic relations used in the experimental set-up and elaborate on the results on both training and test data sets of the SEMEVAL-2007 competition (Girju et al. 2007).

3.1 Data

For semantic type detection, we use 7 binary relations from the training set of the SEMEVAL-2007 competition, all definitions of which share the requirement of the syntactic closeness of the arguments. Further, they have various restrictions on the nature of the arguments. The definitions of the relation types together with the restrictions imposed on them are reproduced below (based on the SEMEVAL-Task 4 definitions).

CAUSE - EFFECT(X,Y) takes place if, given a sentence S, it is possible to entail that X is the cause of Y. X, Y can each be a nominal denoting an occurrence (e.g., event, state, activity), or a noun denoting an entity, as a metonymic expression of an occurrence. In case an effect is caused by a combination of events, each such event is considered a separate cause for the effect. Indirect causation is considered positive, e.g. CAUSE-EFFECT(earthquake, aftershock).

INSTRUMENT - AGENCY(X,Y) is true if the situation described in S entails the fact that X is the instrument of Y (Y uses X). Further, X is an entity and Y is an explicit actor or an implied activity (there exists an activity even if the close context for X and Y includes no verb). The relation is true if the sentence context implies that Y uses X, Y has used X, or X will likely use Y in the future.

PRODUCT - PRODUCER(X,Y) is true if the situation described in S entails the fact that X is a product of Y, or Y produces X. The producer should be actively involved in the process of bringing the product into existence and not just serve as a raw material. The product can be any abstract or concrete object.

ORIGIN - ENTITY(X,Y) is true if the situation described in S entails that X is the origin of Y. Y is the entity derived from the origin. The origin can be

spatial/geographical or material but it should not be actively involved in the process of bringing the entity into existence (e.g., "light bulb"). The entity should not be part of the origin, e.g. "apple seed", when the "seed" is separated from the "apple". In the case of material origin X, X should undergo considerable processing in order to produce Y. A person/company can be identified as origins if they were not involved in the production of the entity. Objects emitting radiation/heat/light are regarded as producers of such emissions, not just origins. An entity can have several origins, and each of them separately will count as an origin. News and information is conveyed, rather than produced, and its source will be the origin.

THEME - TOOL(X,Y) is true if the situation described in S entails the fact that Y (the tool) is (or was) intended (or designed or used) for some kind of action (V-ing, where V is some verb) in which X (the theme) is the thing that is acted upon (the object of the verb V) or the result of the action. X (the theme) should be an object (e.g., "*wine* glass"), an event ("*concert* hall"), a state of being ("*migraine* drug"), an agent ("*artist* award") or a substance ("*water* filtration"). Y (the tool) should be an object (e.g., "migraine *drug*"), an action (e.g., "service *charge*"), an agent ("military *police*"), or a substance ("salad *oil*"). Psychological features are not allowed as tools (e.g., "death *wish*"). The theme and tool must be two completely different and separate things. Plans, missions, strategies, advice, proposals, methods, process, and similar things are not allowed as tools. Requirements, groundwork, foundations, preliminaries, preconditions, and similar things are not allowed as tools for the theme either.

PART - WHOLE(X,Y), where X is part of Y and this relation can be one of the following five types: Place-Area, Stuff-Object, Portion-Mass, Member-Collection and Component-Integral object.

CONTENT - CONTAINER(X,Y) takes place when a sentence S entails the fact that X is (or was) stored (or carried) inside Y. Moreover, X is not a component of Y and can be removed from it. The container must be clearly delineated in space (sea or cloud are locations rather than containers). There is strong preference against treating legal entities (people and institutions) as content. There is weak preference against treating buildings and vehicles as containers.

Table 4.1 shows a number of training/test examples per each relation type and a number of positive instances per relation (+(train) and +(test)) used in the official SEMEVAL-Task 4 competition. All sentences were collected by quering the Web with some hand-written patterns (Hearst 1992, Nakov 2008). It was assumed that using patterns would result in the extraction of not only positive instances, but also negative ones (near misses). We removed examples unannotated with WordNet 3.0 from the training and test data sets while conducting this experiment (column 5). CONTENT - CONTAINER turned out to be the only relation type whose examples are fully annotated. In contrast, PRODUCT - PRODUCER is a relation type with the most information missing (9 examples removed).

Ideally, we would expect constraints to be of the following types:

1. Both arguments have a very specific type

2. One of the arguments is specific, whereas the other allows for a wider range

of semantic types

The third option where both arguments are general does not seem to be appropriate because in such cases it would be difficult to discriminate between different relation types. If we consider such relations as PRODUCT-PRODUCER or CONTENT-CONTAINER, they seem to fall in the second category. For instance, in a typical scenario, a producer would be a human being, while a product can be anything (e.g., a thought, an idea, a table). Similarly, a container in the CONTENT-CONTAINER relation is most likely of the limited type but content may vary substantially.

relation type	all (train)	+(train)	+(test)	+(test, a/w WordNet)
ORIGIN - ENTITY	140	54	81	77
PRODUCT - PRODUCER	140	85	93	84
THEME - TOOL	140	58	71	66
INSTRUMENT - AGENCY	140	71	78	74
PART - WHOLE	140	65	72	71
CONTENT - CONTAINER	140	65	74	74
CAUSE - EFFECT	140	73	80	74

Table 4.1: Distribution of the SEMEVAL examples.

We hypothesize that ORIGIN-ENTITY and THEME-TOOL are the relation types which may require sentential information to be detected. These two relations allow a greater variety of arguments and semantic information alone might be not sufficient. Such relation types as PRODUCT-PRODUCER or INSTRUMENT-AGENCY are likely to benefit more from the knowledge found in ontologies.

3.2 Experiments

To conduct experiments, we collected arguments of all positive examples from the training set and clustered them. In total, there are 14 clustering solutions (2 solutions per relation). An optimal number of clusters is not known in advance and for this reason we set it to 3, 5 and 10. Further, the strength coefficient from Section 2 was used to determine pairs of clusters that cover the training data the best. On the one hand, we may expect a pair of clusters with the strength coefficient 1 to yield 100% precision on the training data but this assumption is, however, misleading. As we use LCS on cluster's elements, it may lead to a very general concept in the WordNet hierarchy and, consequently, to lower precision. On the other hand, if all resulting pairs of clusters are used (as long as the strength coefficient is larger than zero), we should reach 100% recall. The most desirable solution is a pair of clusters that has a high strength coefficient, has a good coverage and whose clusters describe reasonably general concepts.

Selecting the pairs of clusters that should ideally lead to the best performance in the future can be done by by fixing the strength coefficient and by doing so,

restricting ourselves to a subset of clustering pairs. To determine which strength coefficient is the best for a given semantic relation, either the highest accuracy or the F_1 score can be used. The general tendency that one can observe in Figure 2, 3, and 4 is that by increasing the strength coefficient recall increases while precision drops. The X-axis in these figures has to be read as follows. The values there indicate which subset of the entire set of cluster pairs is used. For instance, '0.8' means that all cluster pairs that have the strength coefficient larger than 0.8 are employed. Note that the highest F_1 is not necessarily to be found on the intersection of the precision and recall curves. This happens due to the fact that the strength value of 1.00 does not guarantee the highest precision (as explained above). Table 4.2 shows which values of s were selected per relation and what the performance on the training set is. The solutions with 10 clusters turned out to be the best for all seven relations. Two semantic relations whose scores substantially differ from the rest are THEME - TOOL and ORIGIN - ENTITY.

Relation type	Precision	Recall	Accuracy	s
ORIGIN - ENTITY	56.8	98.0	71.1	wup10, 0.71
CONTENT - CONTAINER	69.2	100	79.6	wup10, all
CAUSE - EFFECT	74.0	100	81.6	wup10, all
INSTRUMENT - AGENCY	84.6	82.1	83.6	wup10, 1.0
PRODUCT - PRODUCER	73.3	94.9	75.4	wup10, 0.29
THEME - TOOL	55.0	88.0	67.4	wup10, 0.29
PART - WHOLE	76.7	87.5	81.8	wup10, 0.5
avg.	69.9	92.9	77.2	

Table 4.2: Performance on the SEMEVAL training data set, where s stands for the strength coefficient.

The results on the test set are given in Table 4.3. Here, we can observe the same tendency as on the training data, namely, CAUSE - EFFECT and INSTRUMENT - AGENCY are among the relations with the highest scores, while THEME - TOOL and ORIGIN - ENTITY belong to semantic relations that cannot be easily identified by using the rules that we derived. It is worthwhile to recall that some constraints are consistent with the part of the test data and not necessarily with all examples. If syntactic context is not used, they are bound to extract false positives. Semantic relations that we consider here can be roughly divided into two groups, where CAUSE - EFFECT, CONTENT - CONTAINER, PRODUCT - PRODUCER, INSTRUMENT - AGENCY, PART - WHOLE form a group of relations that can be relatively easy identified solely on the semantic types of the arguments, whereas ORIGIN - ENTITY and THEME - TOOL cannot. Some constraints found by our method are listed in Table 4.4.

Relation type	Precision	Recall	Accuracy
ORIGIN - ENTITY	51.2	61.8	57.1
CONTENT - CONTAINER	70.3	68.4	68.9
CAUSE - EFFECT	68.4	72.2	70.3
INSTRUMENT - AGENCY	77.8	56.8	70.3
PRODUCT - PRODUCER	73.3	80.0	67.9
THEME - TOOL	30.7	33.3	48.5
PART - WHOLE	58.3	53.8	69.0
avg.	61.4	60.9	64.6

Table 4.3: Performance on the SEMEVAL test data set.

Relation	$(\mathcal{G}_x, \mathcal{G}_y)$
INSTRUMENT - AGENCY	(unit#6, person#1)
	(unit#6, medical_man#1)
CAUSE - EFFECT	(event#1, human_action#1)
	(knowledge#1, human_action#1)
	(event#1, state#2)
	(phenomenon#1, physical_process#1)
PRODUCT - PRODUCER	(object#1, person#1)
	(object#1, group#1)
	(matter#3, physical_entity#1)
	(communication#2, person#1)
ORIGIN - ENTITY	(group#1, object#1)
	(object#1, object#1)
	(object#1, person#1)
THEME - TOOL	(abstract_entity#1, event#1)
	(knowledge#1, abstract_entity#1)
	(event#1, communication#2)
PART - WHOLE	(person#1, group#1)
	(body_part#1, thing#12)
	(substance#1, matter#3)
	(person#1, person#1)

Table 4.4: Some constraints per relation type.

Combining semantic and syntactic evidence

In the previous section, we have shown that if accurately generated, semantic types should provide high precision. In the previous work, we have also noticed that the shortest path introduced by Bunescu and Mooney (2005) usually boosts recall (Katrenko et al. 2010). If we put two pieces of evidence together, we might expect better performance.

Clearly, there are several relations that are likely to benefit from use of semantic types, e.g., CAUSE - EFFECT, INSTRUMENT - AGENCY and PRODUCT -

PRODUCER. For these relations, recall is high (Table 4.5) and this means that accurate constraints would hopefully increase precision without significant decrease in recall. For other relation types, one might attain better performance but this should mostly affect results only slightly because recall by shortest path method is already low.

Relation type	Precision	Recall	Accuracy
ORIGIN - ENTITY	40.0	16.7	51.9
CONTENT - CONTAINER	50.0	23.7	48.6
CAUSE - EFFECT	54.7	100.0	57.5
INSTRUMENT - AGENCY	53.9	92.1	57.7
PRODUCT - PRODUCER	69.9	93.6	68.8
THEME - TOOL	56.3	31.0	62.0
PART - WHOLE	42.9	23.1	61.1
avg.	52.5	54.3	58.2

Table 4.5: Shortest path kernel's results per relation type.

Relation type	Precision	Recall	Accuracy
ORIGIN - ENTITY	57.1	11.1	56.8
CONTENT - CONTAINER	70.0	18.4	54.1
CAUSE - EFFECT	72.1	75.6	72.5
INSTRUMENT - AGENCY	80.0	41.7	70.5
PRODUCT - PRODUCER	77.4	77.4	69.9
THEME - TOOL	50.0	10.3	64.8
PART - WHOLE	60.0	11.5	65.3
avg.	66.7	35.1	64.8

Table 4.6: Results on the SEMEVAL test data set achieved by combining syntactic and semantic evidence.

We use the Stanford parser [1] to analyze data, and combined syntactic and semantic evidence as discussed in Section 2. If semantic types cannot be applied to the test data (because of the lack of annotations with WordNet synsets), predictions by shortest path kernel are used. The results of such combination are presented in Table 4.6. As expected, combination of the pieces of evidence boosts performance for CAUSATION, PRODUCT-PRODUCER and INSTRUMENT-AGENCY but also has a positive influence on other relations in terms of accuracy.

4 Discussion

Using clustering in the method we have proposed requires making some additional decisions, e.g., how many resulting clusters should be considered. To estimate the number of clusters, we use either precision or accuracy on the training data set. Our approach also depends on positive examples in the training set and on the semantic

[1] `http://nlp.stanford.edu/software/lex-parser.shtml`

hierarchy we use. If some parts of the hierarchy are more flat, the resulting patterns may be too general.

Our method has some similarities with the methods by Moldovan et al. (2004) proposed in the past. For instance, we also employ the idea of restricting the WordNet hierarchy, but it is done in the different way and for a different purpose. We do not contrast one semantic relation against the other but rather look for the semantic types of the argument per semantic relation.

Recall also the work by Chaffin and Herrmann (2001) who studied different similarity effects on semantic relations. In contrast to their proposal, we do not rely on similarity between two argument fillers but rather detect similarity on the paradigmatic level. Item similarity (or in our case similarity between two arguments of the same relation) is undoubtedly important for SYNONYMY but we believe that it is nearly useless if other semantic relations are considered.

Semantic constraints and the SEMEVAL guidelines Constraints yielded by our method largely correspond to the relation description given in the guidelines of the SEMEVAL competition. For instance, for CONTENT-CONTAINER containers are often deliniated in space as required by the guidelines. Similarly to what was stated in the SEMEVAL guidelines, we obtained generalizations for several subtypes of MERONYMY. The most successful generalizations were of the type MEMBER-COLLECTION (e.g. <person#1, group#1> in Table 4.4). Most pairs that were judged as good candidates for INSTRUMENT-AGENCY and PRODUCT-PRODUCER relations correspond to the definitions of these relations as well. In particular, the constraints detected for INSTRUMENT-AGENCY have an actor as a type for Agency and an entity for Instrument. In line with the definition of PRODUCT-PRODUCER, products can be either abstract or concrete entities (e.g., <object#1, person#1>). Most generalizations consist of products as concrete entities, there are however examples of abstract entities for the products as well (e.g., <communication#2, person#1>). CAUSATION is one of the relations where almost any type is allowed for its arguments and it can be seen in a variety of generalizations that were detected. The definitions of the ORIGIN-ENTITY and the THEME-TOOL relations do not list the types of arguments that are allowed for this relation but rather their counterexamples (i.e. plans and strategies are not allowed as tools).

Differences among semantic relations It is tempting to conclude that ORIGIN - ENTITY and THEME - TOOL are relations where semantic constraints do not play such a significant role. Indeed, even on the training set, these semantic relations could not be validated with precision that would be comparable to the other relations. Since our method, which exploits only WordNet information, does not seem to be very useful, a question becomes whether context would help more. The best performing system in category "C4" (no WordNet, but queries are used) designed by Nakov (2007) yields much lower scores for these two relations and for PART - WHOLE, which is in line with our results. One explanation for why THEME-TOOL seems to be so different from other relations would be in the way it is defined. Ac-

cording to its definition, two arguments are related to each other via "some kind of action" (Section 3.1). In our view, this relation may be treated as ternary rather than binary.

If used alone, constraints generated by both methods may provide accuracy similar or higher to the existing solutions based on syntactic information such as the shortest dependency kernel. Given performance on the test data set, our method provides rather coarse generalizations which results in a relatively high recall.

5 Conclusions and Future Work

We have presented a method to derive constraints for semantic relations and studied it on seven relations. As can be seen from the experimental results, some relations can be described by such constraints in a relatively precise manner (e.g., INSTRUMENT-AGENCY, PRODUCT-PRODUCER, CONTENT-CONTAINER), while others cannot (e.g., ORIGIN-ENTITY, THEME-TOOL).

There are several directions which might be explored on this topic in the future. One of them includes finding constraints based on probabilistic clustering. In particular, we may use distributional clustering of arguments given information from large data sets. This would enable finding constraints in the cases when data is not annotated with the WordNet synsets. Yet another direction would be incorporating relational similarity.

Acknowledgments

The authors thank the anonymous reviewers and Simon Carter for their comments. This work was carried out in the context of the Virtual Laboratory for e-Science project (www.vl-e.nl). This project was supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and was part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

References

Budanitsky, Alexander and Graeme Hirst (2006), Evaluating WordNet-based Measures of Lexical Semantic Relatedness, *Computational Linguistics* **32** (1), pp. 13–47, MIT Press, Cambridge, MA, USA.

Bunescu, Razvan C. and Raymond J. Mooney (2005), A shortest path dependency kernel for relation extraction, *Joint Conference on Human Language Technology / Empirical Methods in Natural Language Processing (HLT/EMNLP)*, Vancouver, BC.

Chaffin, Roger and Douglas J. Herrmann (2001), Effects of relation similarity on part-whole decisions, *Journal of General Psychology* **115(2)**, pp. 131–139.

Fellbaum, Christiane (1998), *WordNet: An Electronic Lexical Database*, MIT Press.

Girju, Roxana, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret (2007), SemEval-2007 Task 04: Classification of semantic relations between nominals, *ACL 2007*.

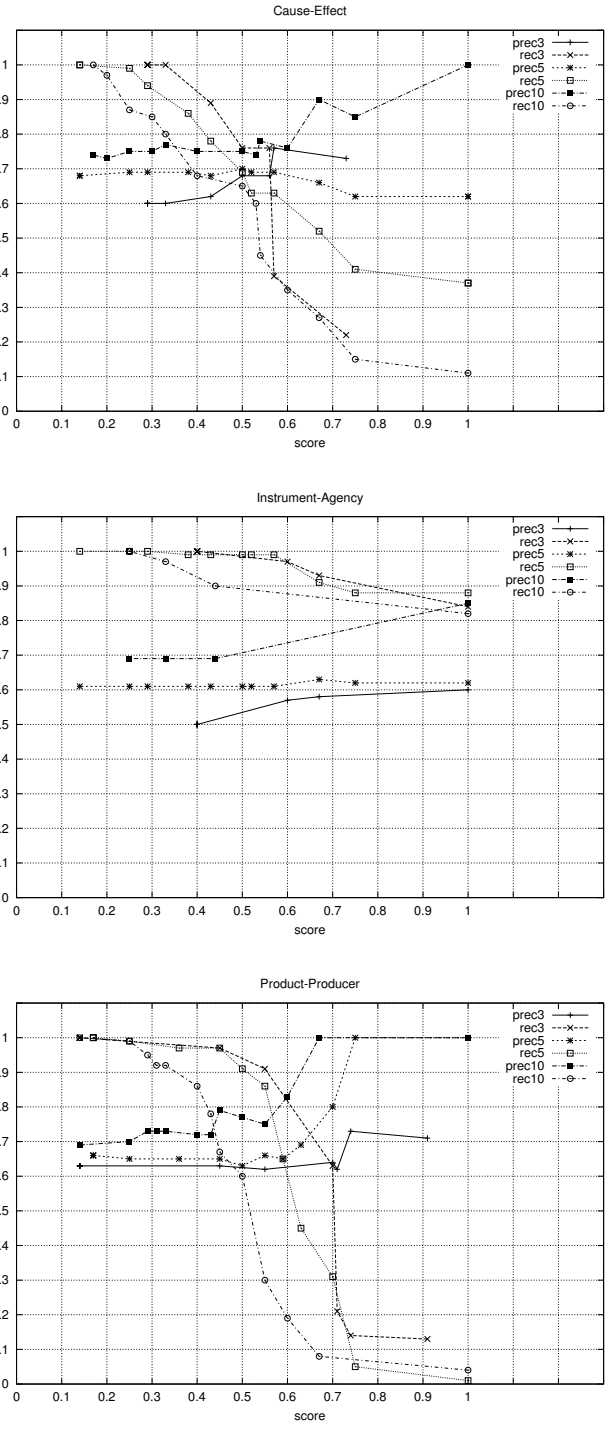


Figure 2: Clustering solutions on CAUSE - EFFECT, INSTRUMENT - AGENCY, and PRODUCT - PRODUCER.

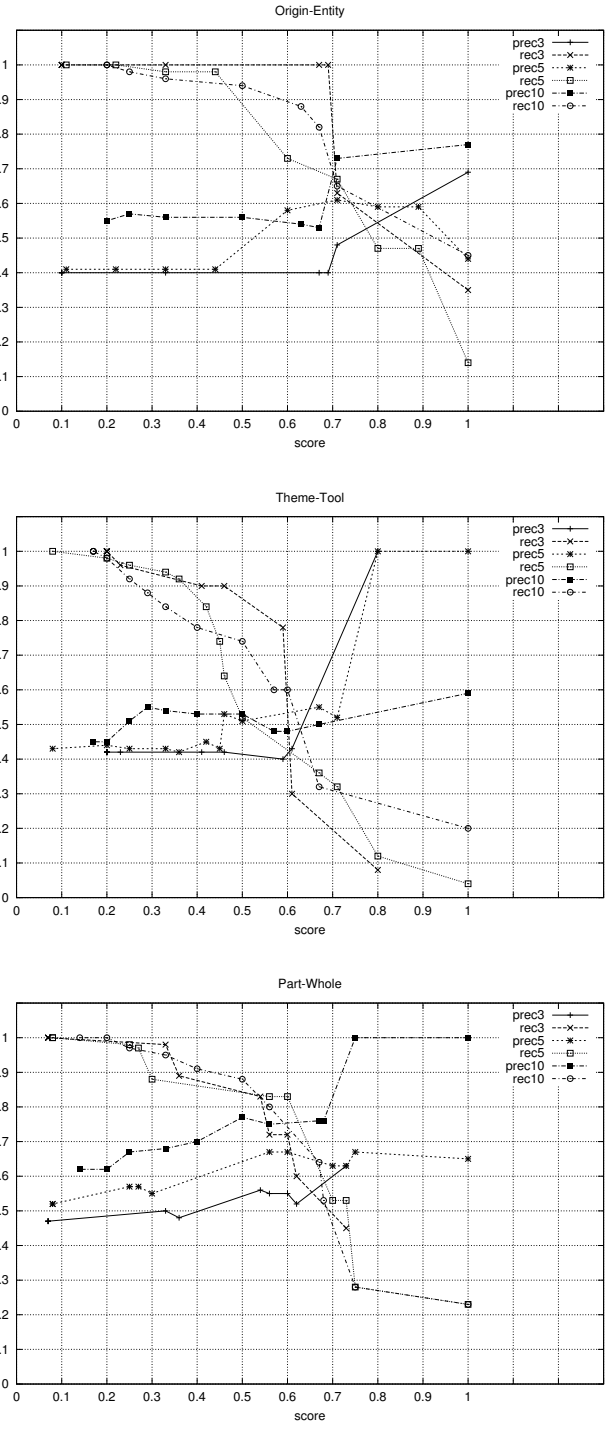


Figure 3: Clustering solutions on ORIGIN - ENTITY, THEME - TOOL, and PART - WHOLE.

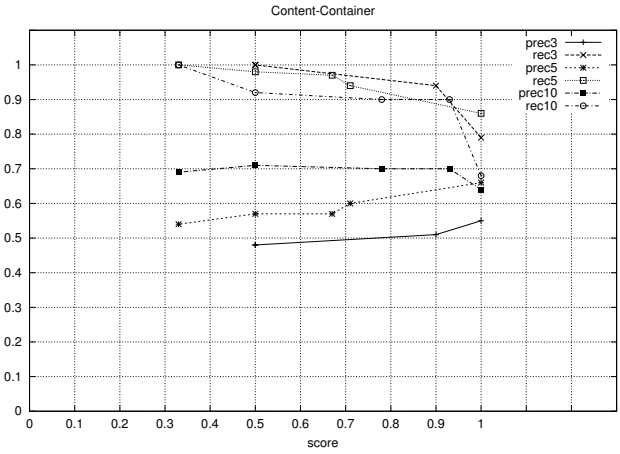


Figure 4: Clustering solutions on CONTENT - CONTAINER.

Hearst, Marti (1992), Automatic acquisition of hyponyms from large text data, *Proceedings of COLING-92*, pp. 539–545.

Katrenko, Sophia, Pieter Adriaans, and Maarten van Someren (2010), Using local alignments for relation recognition, *International Journal of Artificial Intelligence Research (JAIR)* **38**, pp. 1–48.

Khoo, Christopher S. G. and Jin-Cheon Na (2006), Semantic relations in information science, *Annual Review of Information Science and Technology* **40**, pp. 157–228.

McDonald, Ryan (2005), Extracting relations from unstructured text, *Technical Report MS-CIS-05-06*, UPenn.

Moldovan, Dan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju (2004), Models for the semantic classification of noun phrases, *HLT-NAACL 2004: Workshop on Computational Lexical Semantics*, pp. 60–67.

Nakov, Preslav (2007), UCB: System description for SemEval task #4, *SemEval-2007*.

Nakov, Preslav (2008), Paraphrasing verbs for noun compound interpretation, *Proceedings of the Workshop on Multiword Expressions (MWE'08), in conjunction with the Language Resources and Evaluation conference, Marrakech, Morocco, 2008*.

Palmer, Martha and Zhibiao Wu (1995), Verb semantics for English-Chinese translation, *Technical report*, Technical Report No. MS-CIS-95-39, Department of Computer & Information Science, University of Pennsylvania.

van der Plas, Lonneke (2008), *Automatic Lexico-Semantic Acquisition for Question Answering*, PhD thesis, University of Groningen.

Zhao, Ying and George Karypis (2002), Evaluation of hierarchical clustering algorithms for document datasets, *CIKM 2002*.

A Sentence Generator for Dutch

Daniël de Kok and Gertjan van Noord

University of Groningen

Abstract

The paper presents an efficient, wide-coverage, sentence generator for Dutch, which employs the Alpino grammar and lexicon. This generator consists of a chart-based sentence realizer that builds grammatical sentences for a given abstract dependency structure, and a maximum-entropy fluency ranker which selects the most fluent sentence from a set of candidate sentences for a given dependency structure. The coverage, speed and accuracy of the generator is evaluated on several corpora.

1 Introduction

Sentence realizers have been developed for various languages, including English and German. While the generation algorithms used in sentence realizers are very generic, the implementation of a realizer is quite specific to the grammar formalism and input representation. This paper describes a sentence realizer for the wide-coverage Alpino grammar and lexicon.

Alpino (van Noord 2006) is a parser for Dutch which includes an attribute-value grammar inspired by HPSG, a large lexicon, and a maximum entropy disambiguation component. Dependency structures are constructed by the grammar as the value of a dedicated attribute. These dependency structures constitute the output of the parser. Detailed documentation of the dependency structures is given in van Noord et al. (2010).

In generation, the grammar is used in the opposite direction: we start with a dependency structure, and use the grammar to construct one or more sentences which realize this dependency structure. In the general case, a given dependency structure can be realized by more than a single sentence. For instance, the sentence *Na de verkiezingen beklijfden de adviezen echter niet* (*After the elections the advises did, however, not persist.*) is mapped to a dependency structure which can also be realized by variants such as *Na de verkiezingen beklijfden de adviezen niet echter*, or *echter beklijfden na de verkiezingen de adviezen niet*. Therefore, a maximum entropy fluency ranker is part of the generator. The fluency ranker selects the most appropriate, 'fluent', sentence for a given dependency structure.

1.1 Dependency Structures

Various different abstract sentence representations have been proposed as input for sentence realization algorithms, such as Minimal Recursion Semantics (Copestake

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

et al. 2005) and dependency structures (Hays 1964). A useful representation conforms to three characteristics: the representation should be easy to process by external applications; the representation should be abstract enough to map to interesting variation in realizations; and the representation should be native to the grammar and lexicon.

Dependency structures can be argued to conform to these three characteristics. There is plenty previous work based on dependency structures, such as sentence compression (De Kok 2008), machine translation (Lin 2004), and sentence fusion (Marsi and Krahmer 2005), demonstrating its usability. Dependency structure is also native to the Alpino grammar and lexicon.

For the sentence realizer, we use the same dependency structure as created by the Alpino parser, except that we remove information about word order and word inflection. Words are represented in the dependency structure by their root forms, plus optional additional POS-tag information to select for a specific reading. The POS-tag information is presented by attributes such as *pos* with values *verb, noun,* ..., and *num* with values *sg,pl*. Underspecification of such attributes in the input is possible. For instance, leaving out the attribute for number (*num*) for a noun might realize the noun in singular or plural.

2 Sentence realization

2.1 Representation of dependency structures

The attribute-value grammar underlying Alpino constructs dependency structures by means of unification. Dependency structures are represented by attribute-value structures. A category such as *np* has a special attribute *dt* which represents its dependency structure.

The attribute-value structure representation includes information of the head word of that dependency structure, as well as attributes such as *su, obj1, obj2, mod, det* for each of its dependents (subject, direct object, secondary object, modifier, determiner). If there can be multiple dependents of the same type (e.g. for modifiers), a list-valued attribute is used. A special atomic value is used to represent the lack of a dependent of a particular type. For instance, if the input does not contain a direct object dependency, then the value of the attribute *obj1* will be the atom *nil*.

Consider, for example, the dependency structure in Figure 1a. The word *adviezen* (*advises*, represented by the root *advies*) has one dependent, *de* (*the*) with the relation *det*. The word *beklijven* (*to persist*) is represented by the root *beklijf*, and takes the dependency structure associated with *advies* as its subject. This dependency structure is represented by the attribute-value structure shown in Figure 1b. In this structure we only list the attributes which have a value different from the special value *nil*.

Sentences are realized using a bottom-up chart generator. The generator assumes that grammar rules contain the attribute-value structure of the mother node, and a list of attribute-value structures for each of the daughter nodes. Moreover,

$$\begin{bmatrix} \text{HD} & \left[\text{LEX} \quad \textit{beklijf}\right] \\ \text{CAT} & \textit{smain} \\ \text{SU} & \begin{bmatrix} \text{HD} & \left[\text{LEX} \quad \textit{advies}\right] \\ \text{CAT} & \textit{np} \\ \text{DET} & \left\langle \left[\text{HD} \quad \left[\text{LEX} \quad \textit{de}\right]\right] \right\rangle \end{bmatrix} \end{bmatrix}$$

Figure 1: (a) Dependency tree and (b) attribute-value structure for *De adviezen beklijven.*

one of the daughters is identified as the head of the rule (typically a daughter is selected as the head if its dependency structure is equal to the dependency structure of the mother). The chart generator is described in Section 2.2.

In order that the generator constructs partial analyses that are realizing the input dependency structure, top-down guidance is crucial. Top-down guidance requires that every category considered during generation contains a dependency structure which unifies with a part of the input dependency structure. Top-down guidance is explained in more detail in Section 2.3.

During generation, partial realizations are packed in a realization forest for efficiency. Full realizations can be extracted from the packed representation, potentially using the fluency model described in Section 3 to extract only the best possible realization(s). Packing and unpacking are described in Section 2.4 and Section 2.5.

2.2 Chart generation

Chart generation (Shieber 1988, Kay 1996) closely resembles bottom-up chart parsing. A bottom-up chart parser builds derivations bottom-up from the words, chart generation also starts from the words. However, since the ordering of the words is yet to be determined, the worst-case time complexity of chart generation is exponential rather than cubic. This worst-case scenario is observed in practice when we consider modifiers: if a word has k modifiers, often 2^k orderings are admitted by the grammar. Still, chart generation is a relatively efficient algorithm, since partial analyses are constructed only once.

The algorithm schema of chart generation is simple: new edges are put on an agenda. During every iteration, the generator processes one edge from the agenda, and attempts to combine it with edges from the chart. This may lead to new edges. After an edge is processed, it is removed from the agenda and placed on the chart.

An edge represents a grammar rule that is partially or fully completed. Completion here means the process of filling daughter slots. We use two types of edges: *inactive edges* that have all daughters completed and *active edges* that have uncompleted, *active*, daughter slots. An active edge contains the attribute-value structure

of the mother of a rule, and the list of attribute-value structures of the active daughters of a rule. An inactive edge only contains an attribute-value structure for the mother node.

During initialization, all relevant lexical entries are added as inactive edges to the agenda, as well as all mother categories of rules without any daughters (epsilon rules).

When an active edge is processed, the chart is inspected for an inactive edge whose attribute-value structure unifies with the active daughter. Processing an inactive edge involves the same mechanism: find an active edge on the chart such that its active daughter unifies with the inactive edge. In addition, an inactive edge can be used to initialize an active edge if there is a grammar rule with a head daughter that unifies with the inactive edge.

If an edge on the agenda has been processed in all possible ways, it is removed from the agenda and put on the chart. Generation ends when the agenda is empty. Inactive edges with a dependency structure which equals the input dependency structure represent successful realizations.

2.3 Top-down guidance

When chart generation is finished, we are only interested in those derivations where the resulting dependency structure equals the input dependency structure. The input dependency structure constitutes our top-down information. During processing, we only need to construct edges which can be part of a derivation which leads to this input dependency structure.

Since the Alpino grammar is highly lexicalized, top-down information can be enforced in a very efficient manner. The dependency structure of the left hand-side of a rule is built up by unification on the basis of the dependency structures of each of the daughters of the right hand side, in such a way that all dependency information in lexical items ends up as a part of the dependency structure of the top node of the derivation tree. As a consequence, it is possible to 'inject' expected dependency information in the attribute-value structure of the lexical items.

Note that this mechanism is related to Shieber's semantic monotonicity requirement (Shieber 1988), but not identical. In our case, we do assume, as in (Shieber 1988), that the grammar exhibits the monotonicity requirement with respect to dependency structure. But we exploit this requirement one step further: the 'semantics' of a lexical entry is *instantiated* with part of the goal 'semantics'. As a consequence, our bottom-up algorithm is more goal-directed.

As an example, consider the dependency structure given earlier in Figure 1a. During initialization of the chart, lexical look-up is performed with the additional requirement that the value of *dt* unifies with 1b or with a sub-part of 1b. This implies that only the words *de, advies, adviezen* and inflectional variants of the verb *beklijven* are selected during lexical look-up. Moreover, the dependency structure of those lexical entries is already instantiated with parts of the dependency structure of 1b. For instance, the lexical entry for *beklijfden* is given in Figure 2a. Top-down guidance results in the entry given in figure 2b. The example is simpli-

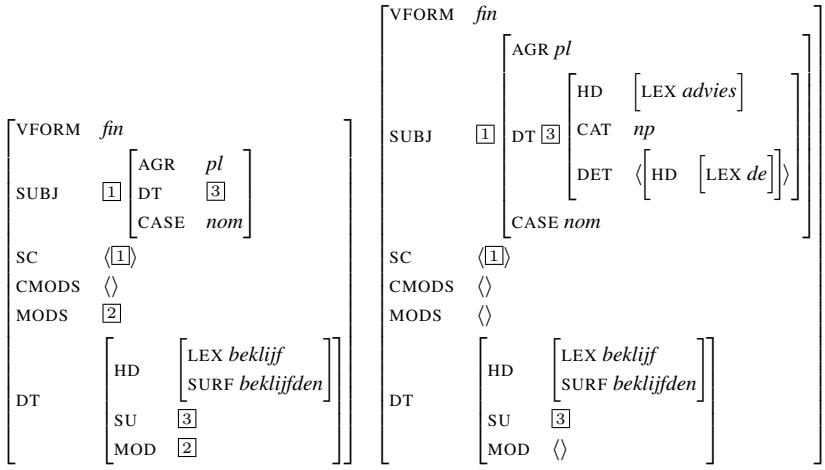


Figure 2: Attribute-value structure for *beklijfden* before (a) and after (b) top-down guidance.

fied for expository purposes.

In Figure 2a, the attribute-value structure states that the subcat list (the attribute SC) of the verb contains a single entry which is identical to the subject (attribute SUBJ). The dependency structure associated with the subject is identical to the SU of the dependency structure of the verb. In addition, the verb can take an arbitrary amount of modifiers. The pair of attribute CMODS and MODS are used to collect those modifiers in a derivation; CMODS contains the list of modifiers found so far, whereas MODS represents the complete list of modifiers.

In Figure 2b, the dependency structure has been instantiated as a result of top-down guidance. Therefore, the verb can only combine with a subject which has a dependency structure associated with *de adviezen*. Moreover, the attribute value structure indicates that there cannot be a single modifier attached to the verb, nor any other dependents.

Top-down guidance not only prevents too many dependencies to be constructed, but it also enforces that all required dependencies are found. For instance, if the input contains a dependency structure of a verb with a subject and a direct object, then lexical look-up will typically not propose an attribute-value structure for the intransitive reading of that verb: that attribute-value structure will have, in the lexicon, the value *nil* for the attribute *obj1* which will not unify with the goal. Similarly, if the input contains a number of modifiers associated with a head, then, typically, maximal projections of that head with fewer modifiers are also ruled out, solving one of the problems raised in (Kay 1996).

Top-down guidance is thus achieved by instantiating each lexical entry with part of the dependency structure of the input. In addition, Shieber's semantic monotonicity requirement is enforced for other edges as well. A new edge is

constructed only in case its dependency structure is unifiable with part of the dependency structure of the goal.

2.4 Packing

For input with a high complexity the generation chart can grow enormously, while there may be many edges with the same attribute-value structure. For instance, the use of optional punctuation does not change an attribute-value structure, while it does introduce new edges for all allowed combinations of punctuation. Another source of growth of the chart are lexical items that have more than one valid inflection.

To compress the chart, we apply *packing*. In packing one inactive edge can represent multiple derivation histories with the same attribute-value structure. The history is represented by simple items, where each item is numbered by the inactive edge representing the item, and contains the identifier of the rule or lexical item used to construct the edge. In the case of a rule we also include a list of pointers to inactive edges that were used to fill the daughter slots of the rule.

For instance, the derivation history `his(31,r(top_start,[23,30]))` represents one particular way the attribute-value structure of inactive edge *31* was constructed, by completion of the *top_start* grammar rule using inactive edges *23* and *30*. Derivation histories representing terminal nodes contain lexical information, such as the root and the Alpino part of speech tag, rather than a list of inactive edge pointers.

Packing is performed when an inactive edge is completed. If an inactive edge is found on the chart with the same attribute-value structure, the history of the new inactive edge is added to the existing inactive edge. We have also experimented with forward-packing (where packing is applied when the new edge is subsumed by an inactive edge on the chart) and backward-packing (where the new edge subsumes an inactive edge on the chart). However, the benefits of both forms of packing did not outweigh the decreased performance caused by subsumption checking.

2.5 Unpacking

After chart generation, full and partial realizations can be retrieved (unpacked) from the packed forest. During unpacking, a derivation tree is created for these derivations. A full realization is represented by an edge with a top category and has a dependency structure that unifies with the target dependency structure. Derivation trees are constructed by expanding histories top-down. The algorithm for expanding a history recursively is shown in the following Prolog fragment:

```
unpack(Id,AttrVal,Tree) :-
    his(Id,His), unpack_his(His,AttrVal,Tree).

unpack_his(r(RuleId,Ds),LHS,tree(LHS,RuleId,Trees)) :-
    grammar_rule(RuleId,LHS,RHS),
    unpack_ds(Ds,RHS,Trees).
```

```
unpack_his(l(Lex),AttrVal,Tree,tree(AttrVal,Lex,[])) :-
    lexical_entry(Lex,AttrVal).

unpack_ds([],[],[]).
unpack_ds([Id|IdT],[AttrVal|AttrValT],[Tree|TreeT]) :-
    unpack(Id,AttrVal,Tree), unpack_ds(IdT,AttrValT,TreeT).
```

The `unpack` predicate retrieves a history with a particular identifier. The auxiliary `unpack_his` predicate performs the actual unpacking. If the history represents a non-terminal, the `unpack_his` predicate applies unpacking to all daughter identifiers. The attribute-value structure of the inactive edge is then reconstructed by retrieving the grammar rule, and completing the rule using the unpacked daughters. If the history represents a lexical node, we retrieve the attribute-value structure for this lexical node, and form a derivation tree leaf node.

Since multiple realizations can generally be generated for a dependency structure, there can be many realizations in the packed forest. Depending on the application, we may want to unpack all or a specified (N) number of realizations. We will use *N-best* unpacking to unpack a specific number of realizations, using a beam that retains only the most fluent realizations for histories representing maximal projections.

3 Fluency ranking

3.1 Introduction

Often many different realizations can be generated for a given input. While all realizations are grammatical, if the grammar is not too permissive, they often differ greatly in fluency. For this reason, we constructed a fluency ranker to select the most fluent realization.

3.2 Model

Different statistical models for fluency ranking have been proposed in the past, such as n-gram language models, maximum entropy models, and support vector machines (Velldal 2008). N-gram language models calculate the probability of a realization purely based on words, while maximum entropy models and support vector machines are linear classifiers that can integrate arbitrary features. Since feature-based models can integrate more information, they perform better than n-gram language models. As Velldal (2008) shows, maximum entropy models perform comparably to support vector machines for fluency ranking, while having a shorter training time. For this reason we use a maximum entropy model in our fluency ranker.

The principle of maximum entropy models is to minimize assumptions, while constraining the expected feature value to be equal to the feature value observed in the training data. In its canonical form, the probability of an event (y) within a context (x) is a log-linear combination of features (f_i) and their weights (λ_i)

(Berger et al. 1996), normalized over all events in that context ($Z(x)$):

$$p(y|x) = \frac{1}{Z(x)} exp \sum_{i=1}^{n} \lambda_i f_i \tag{5.1}$$

The training process estimates optimal feature weights, given the constraints and the principle of maximum entropy. In fluency ranking, a dependency structure is a context, and a realization of that dependency structure is an event within that context.

In fluency ranking, we are not interested in the probabilities of the realizations of a given input, but in the ordering imposed by the probabilities. Since the normalization is constant for every realization given an input, we can simply calculate the linear combination of features and their values to assign a score to a realization:

$$score(y) = \sum_{i=1}^{n} \lambda_i f_i \tag{5.2}$$

3.3 Features

Since a maximum entropy model ranks realizations based on feature values, we have to settle on a set of features that can adequately describe characteristics of fluent sentences. Features for fluency ranking can be divided in two classes: (1) *output features* that describe aspects of the produced sentence, such as the frequency of a word n-gram in the sentence, and (2) *Construction features* that describe aspects of the process that constructed the sentence, such as the frequency of a particular rule being used to derive the sentence. Our current research does not integrate extra-sentential features.

Features can be hand-crafted or created by applying feature templates to training or evaluation data. A feature template can be seen as a function that has a derivation as its input and a set of features as its output. In the following two sections we describe the output and construction features that we use in our fluency ranker.

Output features

Currently, two output features are used, that represent auxiliary distributions (Johnson and Riezler 2000): trigram models for words and part-of-speech tags. For example, consider the sentence *de optische astronomie maakt gebruik van zichtbaar licht* (*the optical astronomy makes use of visible light*), and the corresponding sequence of part of speech tags (*determiner(de)..noun(het,sg,[])*).

We can use the word trigram model to estimate $P_{word}(de..licht)$ and the tag trigram model to estimate $P_{tag}(determiner(de)..noun(het, sg, []))$. The logarithms of these probabilities then become the values of the *ngram_word* and

ngram_tag features, respectively. As shown in Table 5.1, these values are multiplied by the weights of the features that were found during training of the maximum entropy model. If we have no other features, the score of this realization is then the sum of the weighted scores.

Feature	**Weight** (λ_i)	**Value** (f_i)	$\lambda_i \cdot f_i$
ngram_word	0.0158	-62.70	-0.9907
ngram_tag	0.0115	-24.05	-0.2766
Score ($\sum_{i=1}^{n} \lambda_i f_i$)			-1.2673

Table 5.1: Example output feature values for the sentence *de optische astronomie maakt gebruik van zichtbaar licht* (*the optical astronomy makes use of visible light*). Each value multiplied by the feature weight that was found during training of the maximum entropy model. The score of the realization is obtained by summing the weighted feature values.

Both models are trained on newspaper articles from the Twente Nieuws Corpus[1], consisting of 110 million words. For the part-of-speech tag trigram model we use the Alpino part of speech tags.

The probability of unseen trigrams are estimated using linear interpolation smoothing (Brants 2000), where unknown word probabilities are estimated with Laplacian smoothing.

Construction features

We experimented with construction features originating from parse disambiguation, as well as features specifically crafted for fluency ranking. The parse disambiguation features are used in the Alpino parser, and model linguistic phenomena that indicate preferred readings. Phenomena that are modeled include: topicalization of (non-)NPs and subjects; the use of long-distance/local dependencies; orderings in the middle field; identifiers of grammar rules used to build the derivation tree; and parent-daughter combinations.

Furthermore, we use some of features that were devised by Velldal (2008) for fluency ranking. These features describe local derivation sub-trees with optional grandparenting, including variants that contain the binned frequency and standard deviation of the words a sub-tree dominates over.

3.4 Feature selection

Since most features are extracted automatically using feature templates, many features are not necessary in an effective model because they occur only sporadically, correlate strongly with other features in the model (they show the same behavior within specific events), or have little or no correlation to the ranking.

[1] `http://wwwhome.cs.utwente.nl/druid/TwNC/TwNC-main.html`

Feature selection tries to extract $S \subset F$ from a set of features F, such that a model using S performs comparable to a model using F. This is particularly useful if $|S| \ll |F|$, which is true in our experiments where we compress the most effective model from 500,911 features to 2,200 features.

Frequency-based selection has often been used in the literature. In this method features that change often within the same context are selected. However, this approach does not account for overlapping or noisy features (De Kok 2010). For this reason, we use a maximum entropy selection method. In this selection method, a maximum entropy model is built one feature at a time, always selecting the feature the brings the model closest to the training data. To make this computationally tractable, the method assumes that the weights of features that were previously selected are not affected by the addition of a new feature. Since noisy and overlapping features do not contribute to the model, they are not selected.

4 Evaluation methodology and results

4.1 Sentence realizer

We evaluated the sentence realizer using the Alpino test suites and sentences from Wikipedia. These test suites contain sentences, and their manually corrected dependency structures. The test suites are used to detect regressions in the parser, and are now used for the same purpose in generation.

To test the sentence realizer we keep track of the fraction of dependency structures for which realizations could be constructed (coverage), the number of realizations, and the time required for constructing all realizations for a given dependency structure. Using this information, we can extract various interesting characteristics, such as the coverage of the realizer, and the average generation time for a dependency tree of a certain complexity.

We have evaluated the sentence realizer using three Alpino test suites (the so-called *g*, *h*, and *i* suites). These suites where created during grammar development, are generally of increasing complexity, and cover a wide array of lexical and grammatical phenomena. Additionally, we have created a new suite for the evaluation of the sentence realizer, based on sentences of 5 to 25 tokens that were randomly selected from the Dutch Wikipedia of August 2008[2]. This suite was added to evaluate the sentence realizer and fluency ranker on real-world data. Table 5.2 shows the coverage of the sentence realizer for these suites.

As we can see in this table, coverage is complete for the *g* suite, and very good for the *h* and *i* suites. Most of the problematic dependency structures from the *h* and *i* suites contain lexical information that could not be found in the lexicon, nor be handled through the productive lexicon. There are also some less interesting cases, such as empty dependency structures, derived from sentences that only contain punctuation. As expected, the coverage on the Wikipedia suite is lower. In this suite, the generator encounters more unknown roots and noise, such as equations and English phrases.

[2] `http://ilps.science.uva.nl/WikiXML/`

Suite	Inputs	$\geq$ 1 realization	Coverage (%)
g_suite	996	996	100.0
h_suite	991	965	97.4
i_suite	271	262	96.7
Dutch Wikipedia	15398	13701	89.0

Table 5.2: Coverage of the chart generator on various test suites.

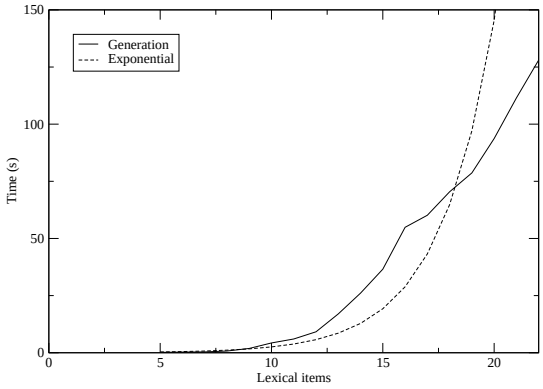


Figure 3: Average time for generating all realizations of dependency structures with a varying number of lexical items.

As described in Section 2.2, the worst-case time complexity of chart generation is exponential. Figure 3 shows the average time for generating all realizations of dependency structures with 5-22 roots, derived from the Wikipedia suite. The figure also shows the exponential after fitting with the average times. We can see that the amount of time required for generation grows enormously with the input complexity, but not yet exponentially.

4.2 Fluency ranking

Generation of data

The training and evaluation data for the fluency ranker was created by parsing the Wikipedia sentences described in Section 4.1 using the Alpino parser. The dependency structure corresponding to the best parse as selected by the disambiguation component was extracted and assumed to be the correct parse. Alpino achieves a concept accuracy of around 90% on common Dutch Corpora (Van Noord 2007).

Furthermore, we assume that the sentence from Wikipedia is the most fluent realization of the extracted dependency structure. This assumption often does not hold in the strict sense. If we are charitable, we can assume that a writer expresses meaning in the most fluent manner, however this does not exclude the possibility that there are multiple fluent realizations.

We then use the chart generator to build the sentences that realize each dependency structure. The derivation trees and associated feature structures for these realizations are compressed, and stored in a derivation tree-bank. We also assign a quality score to each realization by comparing it to the original Wikipedia sentence using the General Text Matcher (GTM) method (Melamed et al. 2003). Finally, we extract the fluency features of all trees in the derivation tree-bank.

4.3 Training of the model

To assess the usefulness of various classes of features, we train fluency ranking models with different sets of features. Each model is trained using 6786 training instances, where each training instance represents a dependency structure, and consists of the quality score and extracted feature values for each realization of that dependency structure. For each training instance, we randomly select 100 realizations to get a representative sample of the training data. A maximum entropy model is trained with a Gaussian prior of 0.001.

Quantitative evaluation

The fluency ranking models are evaluated by applying the rankers to 6785 test cases, counting how often each model chooses the most fluent realization from the set of realizations for a particular dependency structure. The realization with the highest GTM score is considered to be the most fluent realization in the set (*best match*).

Table 5.3 shows an example of this methodology. The table contains the realizations of a dependency structure obtained by parsing the phrase *de geletterdheid van de volledige bevolking wordt geschat op 36%* (*the literacy of the full population is estimated to be 36%*), along with their quality scores, and the fluency scores assigned by the fluency ranking model. If the fluency ranker assigns the highest score to the realization with the highest quality, it picked the most fluent sentence. In our evaluation, we only consider dependency structures with ≥ 5 realizations (5032 instances) to make the task difficult enough.

For the evaluation of the accuracy of fluency ranking models we apply very mild feature selection using a low frequency cut-off, including features that change their value within at least 4 contexts. Such a conservative selection has a negligible impact on the accuracy of the ranker, while making evaluation faster.

We have evaluated various different models consisting of different classes of features. In Table 5.4 we show the best match and GTM scores of the models that we evaluated. The first model (*ngram*) focuses purely on output features, integrating auxiliary distributions of the word and tag trigram models. Since this

Realization	Quality (GTM)	Ranker score
de geletterdheid van de volledige bevolking wordt geschat op 36%	**1.00**	**-1.20**
de geletterdheid van de volledige bevolking wordt op 36% geschat	0.74	-1.56
op 36% wordt de geletterdheid van de volledige bevolking geschat	0.65	-1.66
op 36% wordt de geletterdheid geschat van de volledige bevolking	0.51	-1.81

Table 5.3: A fluency evaluation example for a dependency structure with four realizations. If the fluency ranker assigns the highest score to the realization that has the highest quality score (as in this example), the ranker has picked the most fluent realization.

model is also trained using maximum entropy modeling, weights are assigned to each auxiliary distribution.

Model	Best match (%)	GTM
ngram	33.35	0.6266
ngram + parse	41.41	0.6586
ngram + velldal	43.60	0.6580
all	44.69	0.6633

Table 5.4: Best match accuracies and GTM scores for fluency models incorporating n-gram, parse disambiguation, and Velldal's generation features. The model using a combination of these features outperforms models that do no include one (or more) of these feature sets.

Adding parse disambiguation features (*ngram + parse*) or Velldal's generation features (*ngram + velldal*) to the model improves performance considerably. Here, Velldal's features seem to have more effect, although many of the features extracted using Velldal's templates indirectly describe the same characteristics as the more linguistically-motivated parse disambiguation features. Finally, combining all these feature classes gives the best model.

The best match accuracies in Table 5.4 may seem relatively low, however, these scores should be seen as an indication of the relative performance of each model. Since we created the training and testing data from Wikipedia sentences, there was no manual verification that the dependency structure used was the correct reading of the original sentence. Additionally, we only have the original sentence as an annotation, while there is often more than one fluent sentence. For this reason, the next section describes a qualitative evaluation.

Qualitative evaluation

We have performed a preliminary qualitative evaluation, by manually judging the realization that was selected by the fluency ranker with all features for 100 evaluation instances ourselves. To each selected realization, we assign one of three categories: fluent; neutral (fluent with some minor deficiency, such as missing punctuation or an harmless incorrect inflection); or non-fluent. We found that 80% of the evaluations was fluent, 8% neutral, and 12% not-fluent.

Nearly all of the realizations in the 'neutral' category had a missing comma where it would improve fluency, or an incorrect noun inflection. For efficiency, we currently generate with the minimum amount of punctuation possible. Allowing more punctuation could improve readability. The incorrect noun inflections were produced by the productive lexicon.

Realizations in the 'not-fluent' category are more diverse, but a problem that we often encounter is ordering in coordinations. For instance, the fluency ranker currently has no preference for *de man en zijn hond* (*the man and his dog*) over *zijn hond en de man* (*his dog and the man*), unless a particular case was seen in the training data.

Feature selection

Finally, we compress the best-performing model using feature selection. To compress the model, we first extract the best 10,000 features according to maximum entropy feature selection. We then create models of 100 to 10,000 features with a step size of 100 features. After plotting the accuracies of all models, we pick the amount of features that gives the same accuracy as the baseline model. The baseline model uses mild feature selection (excluding features with values that change within less than 4 contexts).

Method	Features	Accuracy (%)
no selection	500,911	44.83
baseline (cutoff-4)	90,521	44.69
frequency	8,900	44.12
maximum entropy	2,200	44.12

Table 5.5: Accuracies and model sizes after applying feature selection. Maximum entropy selection compresses the model enormously, while giving a accuracy comparable to other models.

As we can see in Table 5.5, only 2,200 features are required to make a model that performs comparably to a model with a fixed frequency-based cut-off. This model is tiny in comparison, and can give good insights in what features are important for fluency ranking (De Kok 2010). We also show the results of a frequency-based selection that picks the N most frequently changing features. This method results in a model that is more than four times larger than applying maximum

entropy feature selection.

5 Conclusion

In this paper we have presented a sentence generator that leverages the Alpino grammar and lexicon to provide high-coverage realization from dependency structures. The sentence realizer uses a novel approach to top-down guidance, where information about expected and absent dependency relations is added to the dependency structure of lexical items. This ensures that only the required dependencies are present in derivations. The sentence generator also contains a fluency ranker which attempts to select the most fluent realization for a dependency structure.

In the future, we hope to improve the performance of the generator on very complex dependency structures. For instance, we are currently refining and experimenting with N-best extraction, for applications where only the most fluent realization is required. Additionally, we hope to add more linguistically inspired fluency ranking features to replace the Velldal (2008) features.

References

Berger, A.L., V.J.D. Pietra, and S.A.D. Pietra (1996), A maximum entropy approach to natural language processing, *Computational linguistics* **22** (1), pp. 71, MIT Press.

Brants, T. (2000), TnT – a statistical part-of-speech tagger, *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

Copestake, A., D. Flickinger, C. Pollard, and I.A. Sag (2005), Minimal recursion semantics: An introduction, *Research on Language & Computation* **3** (4), pp. 281–332, Springer.

De Kok, D. (2008), *Headline generation for Dutch newspaper articles through transformation-based learning*, Master's thesis, University of Groningen.

De Kok, D. (2010), Feature selection for fluency ranking, *Proceedings of the 6th International Natural Language Generation Conference*, pp. 155–163.

Hays, D.G. (1964), Dependency theory: A formalism and some observations, *Language* **40** (4), pp. 511–525, Linguistic Society of America.

Johnson, M. and S. Riezler (2000), Exploiting auxiliary distributions in stochastic unification-based grammars, *Proceedings of the 1st NAACL conference*, pp. 154–161.

Kay, M. (1996), Chart generation, *Proceedings of the 34th annual meeting on ACL*, ACL, pp. 200–204.

Lin, D. (2004), A path-based transfer model for machine translation, *Proceedings of the 20th COLING conference*, ACL, p. 625.

Marsi, E. and E. Krahmer (2005), Explorations in sentence fusion, *Proceedings of the European Workshop on Natural Language Generation*, pp. 8–10.

Melamed, D., R. Green, and J. Turian (2003), Precision and recall of machine translation, *HLT-NAACL*.

Shieber, S. (1988), A uniform architecture for parsing and generation, *Proceedings of the 12th COLING conference*, Budapest.

van Noord, G. (2006), **A**t **L**ast **P**arsing **I**s **N**ow **O**perational, *TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles*, Leuven, pp. 20–42.

Van Noord, G. (2007), Using self-trained bilexical preferences to improve disambiguation accuracy, *Proceedings of the 10th International Conference on Parsing Technologies*, ACL, pp. 1–10.

van Noord, G., I. Schuurman, and G. Bouma (2010), Lassy syntactische annotatie, revision 17566.

Velldal, E. (2008), *Empirical Realization Ranking*, PhD thesis, University of Oslo, Department of Informatics.

Extraction of Biomedical Events

Roser Morante, Vincent Van Asch, and Walter Daelemans

CLiPS - University of Antwerp

Abstract

In this paper we describe a memory-based machine learning system that extracts biomedical events from texts relying on information from contextual and syntactic features. The main characteristics of the system are that it uses information from dependency syntax and that it integrates classifiers that learn event triggers and event participants jointly. The results show that this system is more efficient than a similar memory-based system that used shallow context information and integrated classifiers in a traditional pipeline architecture.

1 Introduction

In recent years, research on biomedical text mining has seen substantial progress, (Krallinger and Valencia 2005, Ananiadou and McNaught 2006, Krallinger et al. 2008a). The focus on extraction of event frames using machine learning techniques is relatively new, since most research was devoted to named entity recognition. This was due in part to the lack of annotated corpora. Recently, some corpora have been annotated with event level information of different types: PropBank-style frames (Wattarujeekrit et al. 2004, Chou et al. 2006), frame independent roles (Kim et al. 2008, Pyysalo et al. 2007), and specific roles for certain event types (Sasaki et al. 2008). Thanks to these efforts it is possible now to extract biomedical events by applying machine learning techniques.

Most work on biomedical information extraction focuses on extracting relations between biomedical entities within a text. For example, Bundschus et al. (2008) developed a system to identify relations between genes and diseases from a set of Gene Reference Into Function phrases. Progress in this field has been boosted by the shared tasks on protein-protein interaction extraction in the framework of the Language Learning in Logic Workshop 2005 (Nédellec 2005) and the BioCreative competitions (Krallinger et al. 2008b). Event extraction has emerged to satisfy new information extraction needs. Research on event extraction has benefited from the data and tools made available for the BioNLP Shared Task on Event Extraction 2009 (BioNLP-ST) (Kim et al. 2009), which consisted of finding event triggers and event participants.

In this paper we describe a machine learning system that extracts event triggers and event participants from biomedical texts. The system has been trained and tested on the BioNLP-ST. Although the approach is possible using any classification-based supervised learning method, we chose for Memory-Based Learning (MBL) as learning method. Memory-based language processing

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

(Daelemans and van den Bosch 2005) is based on the idea that NLP problems can be solved by reuse of solved examples of the problem stored in memory. Given a new problem, the most similar examples are retrieved, and a solution is extrapolated from them.

The originality of the system that we present lies in the fact that event triggers and participants are learned jointly, whereas the machine learning systems that were submitted to the task first learn the event triggers, and then the event participants. It has been shown that jointly learning two tasks can lead to better results than learning the tasks apart in a cascade. Wang et al. (2008) jointly learn Chinese word segmentation, named entity recognition, and part-of-speech tagging, outperforming a pipeline architecture baseline. Finkel and Manning (2009) show that joint learning of parsing and named entity recognition produce mildly improved performance for both tasks. Our goal in this paper is to investigate whether the joint setting is suitable for event extraction.

In Section 2 we briefly describe the data and the task. Section 3 introduces the system architecture. Sections 4 and 5 present the system in detail, and Section 6 the results. Finally, some conclusions are put forward in Section 7.

2 Data and task description

The event extraction system that we present has been trained and evaluated with the data provided by the BioNLP-ST[1], which consisted of extracting bio-molecular events from texts, focusing on events involving proteins and genes. The system was developed after the competition finished. In this task, an *event* is defined as a relation that holds between one or more *entities* that fulfill different roles. There are two types of entities: proteins and biomedical events. Entities can be either *participants* or *arguments* of an event. Participants fulfill the core roles (Theme, Cause) in the event, and arguments (Location, Site) further specify the events. In Sentence (6.1), the proteins *STAT1, STAT3, STAT4, STAT5a,* and *STAT5b* are participants of the biomedical event *phosphorylation*. They fulfill the role *Theme*. *Tyrosine* is an argument of the same event, and it fulfils the role *Site*.

(6.1) IFN-alpha enhanced tyrosine phosphorylation of STAT1, STAT3, STAT4, STAT5a, and STAT5b

For this sentence, the task consists of identifying *phosphorylation* as an event trigger, and extracting the five events listed in Table 6.1.

The event types annotated in the corpora are: Gene Expression, Localization, Phosphorylation, Protein Catabolism, Transcription, Binding, and (Positive, Negative) Regulation. All of them have one *Theme* as participant, except for Binding that can have more than one. Regulations have also a participant *Cause*. Sentence (6.2) contains triggers (*binding*) of Binding events with multiple Themes. The events to be extracted are listed in Table 6.2.

[1] Web page: `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/`

	Theme	Argument		Theme	Argument
Event 1	STAT1	tyrosine	Event 4	STAT5a	tyrosine
Event 2	STAT3	tyrosine	Event 5	TAT5b	tyrosine
Event 3	STAT4	tyrosine			

Table 6.1: Events to be extracted from Sentence (6.1).

(6.2) When we analyzed the nature of STAT proteins capable of binding to IL-2Ralpha, pim-1, and IRF-1 GAS elements after cytokine stimulation, we observed IFN-alpha-induced binding of STAT1, STAT3, and STAT4, but not STAT5 to all of these elements.

	Theme(s)	Argument		Theme(s)	Argument
Event 1	STAT4, IRF-1	GAS elements	Event 7	IL-2, Ralpha	GAS elements
Event 2	STAT3, IL-2Ralpha	GAS elements	Event 8	pim-1	GAS elements
Event 3	STAT3, IRF-1	GAS elements	Event 9	STAT1, IRF-1	GAS elements
Event 4	STAT4, pim-1	GAS elements	Event 10	STAT3, pim-1	GAS elements
Event 5	STAT1, IL-2Ralpha	GAS elements	Event 11	IRF-1	GAS elements
Event 6	STAT4, IL-2Ralpha	GAS elements	Event 12	STAT1, pim-1	GAS elements

Table 6.2: Events to be extracted from Sentence (6.2).

Events can be single or nested. A nested event has as argument another event, like in Sentence (6.3), where *effects* triggers a Regulation event, which has as a participant the Gene Expression event *production*.

(6.3) We have studied the effects of prednisone (PDN) on the production of cytokinase (IL-2, IL-6, TNF-alpha, IL-10).

The training corpus provided for the task consists of 176,146 words and 8,597 events, the development corpus of 33,937 words and 1,809 events, and the test corpus of 57,367 words and 3,182 events. The corpora are annotated with gold standard protein and named entities. The task consists of two subtasks: 1) detecting the triggers of events, that is, the words that express the event, and the triggers of participants and arguments that are not proteins; 2) detecting participants and arguments per event.

The system is evaluated in terms of precision, recall and F1 using the evaluation scripts of the BioNLP-ST. We provide intermediate results of the system based on the development data and we provide the final results on development and test data. As in the official results of the BioNLP-ST, here we will report results under the mode Approximate Span Matching and Approximate Recursive Matching (ASM/ARM) as defined in (Kim et al. 2009). In ASM mode, the requirement of exactly matching the text span of triggers is relaxed. The ARM mode relaxes the requirement for recursive event matching, so that an event can be correct even if

the events it refers to are only partially correct. More details about the task setting, corpora, and evaluation can be found in the webpage of the task and in Kim et al. (2009).

3 System architecture

The architecture of the system is represented in Figure 1. It works in three phases: preprocessing, classification and postprocessing.

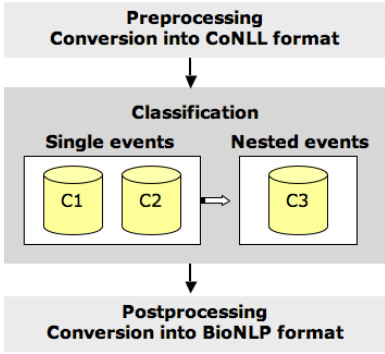


Figure 1: Architecture of the event extraction system.

The system starts by preprocessing the corpora into a learnable format. In order to get information for feature construction for the machine learner, we process the corpora with the GDep dependency parser (Sagae and Tsujii 2007), which outputs for every word the part-of-speech (POS) tag, the lemma, IOB-style chunks, named entities, the syntactic head, and the dependency relation. The data are converted into a column format, following the standard format of the CoNLL Shared Task 2006 (Buchholz and Marsi 2006). Table 6.3 shows a simplified example of a preprocessed sentence. The first column contains the token number in the sentence; the second the word; the third to the sixth contain information provided by the GDEP parser; the seventh column contains the named entities as provided by the BioNLP-ST. The eighth and ninth columns contain as many slots separated by ":" as there are events in the sentence; the eighth column marks with the event type the tokens that are event triggers, and the ninth column marks the tokens that are event participants in the same slot as the event to which they belong. For example, Token 4 "alter" expresses two Regulation events. The first one has as participant Token 9, and the second one Token 11. Triggers are expressed with BI tags (Beginning, Inside) in order to capture multiword triggers.

Rewriting the development corpus into the column format and converting it back into the BioNLP-ST format with gold standard information results in an F1-score of 93.57 %. This score effectively constitutes an upperbound for a machine learner using this data format. The loss in performance can be attributed to the fact

that we do not process intersentential event participants(they amount to 5% of the data), and to the complicated structures of the nested events.

#	WORD	POS	NE	DEP	LABEL	NE TASK	EVENT TRIGGERS	PARTICIPANTS
1	RCC-S	NNS	O	2	SUB	O	_:_:_:_:_	_:_:_:_:_
2	did	VBD	O	0	ROOT	O	_:_:_:_:_	_:_:_:_:_
3	not	RB	O	2	VMOD	O	_:_:_:_:_	_:_:_:_:_
4	alter	VB	O	2	VC	O	B-Reg:B-Reg:_:_:_	_:_:_:_:_
5	the	DT	O	7	NMOD	O	_:_:_:_:_	_:_:_:_:_
6	cytoplasmic	JJ	O	7	NMOD	O	_:_:_:_:_	_:_:_:_:_
7	levels	NNS	O	4	OBJ	O	_:_:_:_:_	_:_:_:_:_
8	of	IN	O	7	NMOD	O	_:_:_:_:_	_:_:_:_:_
9	RelA	NN	B-protein	11	NMOD	B-Protein	_:_:_:_:_	Theme:_:_:_:Theme
10	and	CC	O	11	NMOD	O	_:_:_:_:_	_:_:_:_:_
11	NF-kappaB1	NN	B-protein	8	PMOD	B-Protein	_:_:_:_:_	_:Theme:_:_:Theme:_
12	but	CC	O	2	VMOD	O	_:_:_:_:_	_:_:_:_:_
13	did	VBD	O	2	VMOD	O	_:_:_:_:_	_:_:_:_:_
14	suppress	VB	O	13	VC	O	_:_:B-NegReg:B-NegReg:_:_	_:_:_:_:_
15	their	PRP$	O	17	NMOD	O	_:_:_:_:_	_:_:_:_:_
16	nuclear	JJ	O	17	NMOD	B-Entity	_:_:_:_:_	_:_:_:_ToLoc:ToLoc
17	localization	NN	O	14	OBJ	O	_:_:_:_:B-Loc:B-Loc	_:_:Theme:Theme:_:_
18	and	CC	O	2	VMOD	O	_:_:_:_:_	_:_:_:_:_
19	inhibited	VBD	O	2	VMOD	O	_:_:_:_:_	_:_:_:_:_
20	the	DT	O	21	NMOD	O	_:_:_:_:_	_:_:_:_:_
21	activation	NN	O	19	OBJ	O	_:_:_:_:_	_:_:_:_:_
22	of	IN	O	21	NMOD	O	_:_:_:_:_	_:_:_:_:_
23	RelA	NN	B-protein	27	NMOD	B-Protein	_:_:_:_:_	_:_:_:_:_
24	/	SYM	I-protein	27	NMOD	O	_:_:_:_:_	_:_:_:_:_
25	NF-kappaB1	NN	I-protein	27	NMOD	B-Protein	_:_:_:_:_	_:_:_:_:_
26	binding	NN	I-protein	27	NMOD	O	_:_:_:_:_	_:_:_:_:_
27	complexes	NNS	I-protein	22	PMOD	O	_:_:_:_:_	_:_:_:_:_
28	.	.	O	2	P	O	_:_:_:_:_	_:_:_:_:_

Table 6.3: Example sentence represented in CoNLL format.

The classification component will be described in Section 4. The postprocessing component consists of three rules that are applied to the output of the classification in order to reconstruct the data into the original BioNLP-ST files. Further details are presented in Section 5.

4 Classification of event triggers and participants

The classification component consists of three memory-based classifiers. Two classifiers are used to find triggers and participants of single events, and one classifier is used to find triggers and participants of nested events, based on the output of the first two classifiers.

The algorithm used is TRIBL as implemented in TiMBL (version 6.1.2) (Daelemans et al. 2007). TRIBL is a hybrid combination of IB1, a k-NN classifier, and IGTREE, a fast decision-tree approximation of k-NN (Daelemans and van den Bosch 2005) that splits the classification of instances into a quick decision-tree traversal based on the first, most informative, features, followed by a slower k-NN classification based on the remaining features. In this case the classifier used the two most informative features for the IGTREE classification. The k-NN classifier is IB1, the usual memory-based algorithm based on the k-nearest neighbor classification rule.

Parameterisation of the classifiers was performed by experimenting with sets of parameters on the development set. TRIBL was parameterised in this case by using gain ratio for feature weighting, overlap as distance metric, 5 nearest neighbors for extrapolation, and normal majority voting for class voting weights. The three

classifiers use the same parameters.

In the next subsections the three classifiers are described and intermediate results are provided.

4.1 Learning single events

Two classifiers are used to find triggers of single events and their participants. Instances represent combinations of tokens that are tagged as Proteins and all the tokens in the sentence with POS verb, noun or adjective, which amount to almost 99% of the events in the training corpus. The three classifiers learn the combined label "EventType:ParticipantType".

Classifier 1 (C1) processes the instances in which the combined token is an ancestor of the Protein in the dependency tree, and Classifier 2 (C2) processes the rest of the cases. This is motivated by the fact that the predictive power of features is different, as will be explained below. For the sentence in Table 6.3, C1 processes the combinations of protein *RelA* (Token 9) with Tokens 11, 7, 4, and 2, and C2 processes the combination of the same protein with tokens 1, 6, 9, 13, 14, 16, 17, 19, 21, 23, 25, 26, and 27.

We experimented successfully with the features that we list below. Each group of features is tagged with an identifier in bold characters that will be used in the Tables below where we analyze the performance of the classifier per group of features. We mark with (*) the features that are not used by C1.

- Information about protein and combined token: **[Basic]** word, lemma, POS tag, chunk tag, named entity (NE) tag, and dependency label.
- Information about the context of protein and combined token: **[Context]** the same features as for protein and combined token, for a window of three tokens to the right and three to the left in the sequence of tokens, and n-grams of 1, 2, and 3 tokens for the same window of tokens.
- Information about the path in the dependency tree:

  [Path] Feature indicating who is the ancestor in the dependency tree (protein, combined token, none)*; boolean feature indicating whether combined token is head of protein, number of steps up from protein; number of steps from common ancestor to combined token* if there is a common ancestor; chains of lemmas, POS, and dependency labels from protein to common ancestor if there is one, or to combined token for C1, and from common ancestor, if there is one, to combined token*;

  [Path 2] POS, lemma and dependency label of the tokens that are three steps up from protein and three steps up from the combined token, and from the tokens that are three steps down from the common ancestor in the direction of the protein or from combined token in C1, and three steps down from the common ancestor to the combined token.

- Information about the head and children of protein: **[ProtFam]** Word, lemma, POS, chunk tag NE and dependency label of head, and strings of POS, chunk tags, NE and dependency labels of the children.
- Information about the head and children of combined token: **[EvFam]** the same as for head and children of protein.

Table 6.4 shows the F1 scores of C1 per event type. The number under the event type expresses the frequency of the class.

Features	Event type								
	Bin	GEx	Loc	Pho	PrC	Reg	+Reg	-Reg	Trans
	195	260	39	35	12	37	127	43	64
Basic	53.97	77.30	75.32	80.95	84.61	25.53	34.86	25.45	61.65
+Context	58.40	78.26	74.35	82.92	84.61	**27.45**	47.34	31.74	60.86
+Path	61.29	79.41	74.35	80.95	**91.66**	26.41	50.19	**34.92**	**65.11**
+Path 2	65.94	**79.85**	80.00	82.35	**91.66**	20.00	53.84	22.22	64.12
+ProtFam	**66.66**	79.77	**80.95**	**83.33**	**91.66**	20.83	**54.26**	21.87	64.12
+EvFam	65.39	77.26	79.51	**83.33**	**91.66**	26.92	**54.26**	19.67	61.31

Table 6.4: F1 results of C1 on the development set in CoNLL format. "Bin": Binding; "GEx": Gene Expression; "Loc"': Localization; "Pho": Phosphorylation; "PrC": Protein Catabolism; "Reg": Regulation; "+Reg": Positive Regulation; "-Reg": Negative Regulation; "Trans": Transcription.

As expressed by the "+" character in the first column, features are successively added to the Basic features. We observe that the scores per groups of features are not homogeneous for all event types. It is difficult to find a combination of features that increases the performance for all event types, and it is also difficult to evaluate the scores for the classes that are not frequent. The final system incorporates the version of the classifier that uses the groups of features +ProtFam. The motivation is that these features score the highest for Binding and Positive Regulation, and score only 0.12 lower than the highest for Gene Expression, which are the most frequent event types. These features score also the highest for Localization, Phosphorylation, and Protein Catabolism.

The behavior of features for C1 contrasts with the behavior of features for C2, the results of which are shown in Table 6.5. C2 processes a bigger and more imbalanced data set. In this case, we observe two main characteristics: the Basic features score clearly lower than the highest scoring combination of features, and adding the context features produces lower scores for most event types. When Context features are used, the two most informative features, which TRIBL uses to split the classification, were features about the second and third token to the right of Combined Token, whereas when Context features are not used TRIBL uses the lemma and the word of the Combined Token. Additionally, we also observe that the scores of C2 are much lower than the scores of C1, suggesting that it is more difficult to classify instances in which Protein and Combined Token do not have a dependency relation. The final system incorporates the version of C2 that uses

the groups of features Basic, Path, Path 2, and ProtFam. In general, the most informative features are lemmas of the event context in the sequence of words and in the dependency tree.

Features	Event type								
	Bin	GEx	Loc	Pho	PrC	Reg	+Reg	-Reg	Trans
	102	84	14	11	9	24	51	15	18
Basic	20.15	22.01	12.50	36.36	36.36	0.00	12.90	0.00	0.00
+Context	38.09	6.66	0.00	0.00	0.00	13.33	3.84	0.00	0.00
+Path	3.80	6.66	0.00	0.00	0.00	13.79	7.54	0.00	0.00
+Path 2	3.80	6.59	0.00	0.00	0.00	**14.81**	7.54	0.00	0.00
+ProtFam	3.80	6.66	0.00	0.00	0.00	13.79	7.54	0.00	0.00
+EvFam	3.80	6.66	0.00	0.00	0.00	14.28	7.54	0.00	0.00
Basic	20.15	22.01	12.50	36.36	36.36	0.00	12.90	0.00	0.00
+Path	33.96	29.41	22.22	**43.47**	50.00	11.76	13.33	0.00	0.00
+ Path 2	38.50	32.11	**30.00**	38.09	**53.33**	10.52	**31.16**	0.00	8.00
+ProtFam	**41.46**	33.56	19.04	36.36	40.00	9.52	25.00	0.00	8.33
+EvFam	37.28	**35.21**	19.04	34.78	40.00	9.75	23.68	0.00	**9.09**

Table 6.5: F1 results of C2 on the development set in CoNLL format.

The results of evaluating the system with only C1 and C2 are shown in Table 6.6[2]. Binding and Regulation events score lower, which can be expected because of the possibility of multiple Themes for Binding events and the nested events in Regulations. A positive aspect of these results is that precision is reasonably high.

	Total	Precision	Recall	F1
Binding	248	43.75	28.23	34.31
Gene Expression	356	75.08	63.76	68.96
Localization	53	72.09	58.49	64.58
Phosphorylation	47	65.45	76.60	70.59
Protein Catabolism	21	93.33	66.67	77.78
Transcription	82	66.15	52.44	58.50
Regulation	169	28.00	4.14	7.22
Positive Regulation	617	53.42	12.64	20.45
Negative Regulation	196	26.09	3.06	5.48
TOTAL	1789	61.34	28.62	39.03

Table 6.6: Evaluation of the system with C1 and C2 (ASM/ARM) on development data.

[2]The total number of events in Table 6.6 is not equal to the sum of the total number of events in Table 6.4 plus the total number of events in Table 6.5 because Tables 6.4 and 6.5 count the events in the CoNLL representation, whereas Table 6.6 counts the events in the BioNLP-ST format. The conversion into CoNLL format is not perfect, as indicated in Section 3.

4.2 Learning nested events

In order to find nested events, we add Classifier 3 (C3). Instances represent combinations of a protein or an event predicted by C1 and C2, and all the tokens in the sentence with POS verb, noun or adjective. Some features used by C3 are different from the ones used by C1 and C2:

- **[Lemmas]** Lemmas of the protein/predicted event and of the combined token.
- **[Basic]** Word, POS tag, chunk tag, named entity (NE) tag, and dependency label of the protein/predicted event and of the combined token.
- **[Context]** of the protein/predicted event and of the combined token. The same features as [Basic], for a window of three tokens to the right and three to the left in the sequence of tokens.
- **[Path]** Same as for C1 and C2.

Table 6.7 shows the results of C3 on development data. In contrast with C1 and C2, the Basic features do not achieve results comparable to the highest scores and the +Path features provoke a clear increase in the scores for all event types, yielding the best combination of features.

Features	Event type								
	Bin	GEx	Loc	Pho	PrC	Reg	+Reg	-Reg	Trans
	297	344	55	46	21	101	347	121	82
Lemmas	17.74	33.94	51.61	40.00	63.15	20.28	29.25	6.89	16.07
+Basic	34.00	38.53	52.42	54.11	61.11	16.26	19.65	10.44	27.82
+Context	44.44	59.84	46.15	61.01	77.55	18.42	39.80	22.98	48.97
+Path	**56.88**	**71.84**	**68.68**	**73.58**	**85.71**	**34.48**	**53.37**	**39.19**	**55.69**

Table 6.7: F1 results of C3 on the development set in CoNLL format.

The most informative features for this classifier are: the feature indicating what is the ancestor; the lemma and word of combined token; the string of lemmas from protein to common ancestor or to combined token in the dependency tree; the full form of protein; the number of steps down from common ancestor to combined token, or from protein if protein is the ancestor; the lemmas of the token to the left and one and two tokens to the right of combined token; the dependency label of combined token.

5 Postprocessing

The Classification phase produces a multicolumn file, as shown in Table 6.8. Some heuristics are needed to rewrite the multicolumn file into the original BioNLP-ST format. We defined three rules: multiple events rule, multiple themes rule, nested events rule.

token	word	event trigger 1	participants 1	event trigger 2	participants 2
1	We	–	–	–	–
2	have	–	–	–	–
3	studied	–	–	–	–
4	the	–	–	–	–
5	effects	Regulation	–	–	–
6	of	–	–	–	–
7	prednisone	–	–	–	–
8	(	–	–	–	–
9	PDN	–	–	–	–
10	)	–	–	–	–
11	on	–	–	–	–
12	the	–	–	–	–
13	production	–	Theme	Gene_expression	–
14	of	–	–	–	–
15	cytokinase	–	–	–	–
16	(	–	–	–	–
17	IL-2	–	–	–	Theme
18	,	–	–	–	–
19	IL-6	–	–	–	Theme
20	,	–	–	–	–
21	TNF-alpha	–	–	–	Theme
22	,	–	–	–	–
23	IL-10	–	–	–	Theme
24	)	–	–	–	–

Table 6.8: Output of the Classification component.

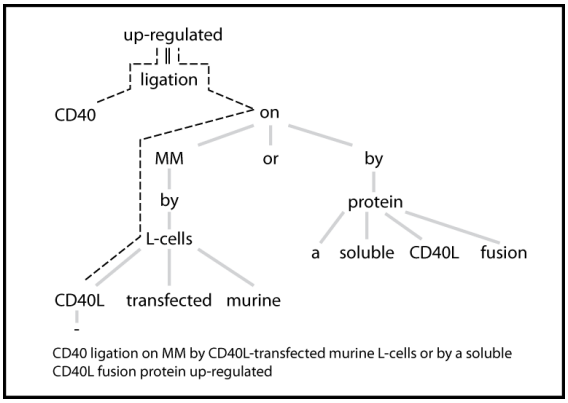
5.1 The multiple events rule

For the sentence in Table 6.8, two events have been predicted. Event 1 expressed by Token 5 is a Regulation event with Event 2 (Token 13) as Theme. Event 2 is a Gene Expression event that has four Themes (Tokens 17, 19, 21, 23). However, the task specifications indicate that a Gene Expression event can have only one Theme. In order to rewrite the output, the system reads in all tokens and the associated predictions containing information about event triggers and participants. One event is created for every predicted trigger and it is assigned a unique identification number. When more than one Theme is predicted for an event, this event is duplicated. As for the example in Table 6.8, the Gene Expression event with four Themes will be split up into four Gene Expression events, each with its own Theme.

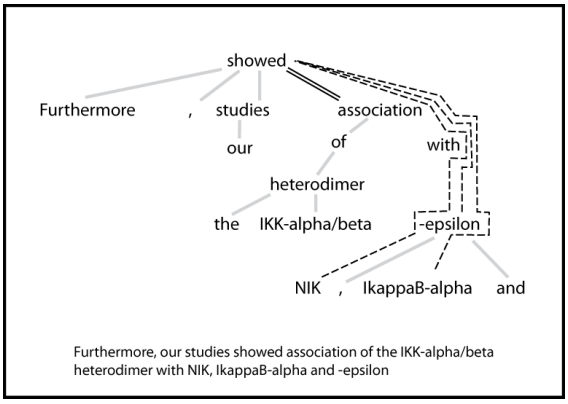
5.2 The multiple themes rule

Binding events can have multiple Themes. By analyzing the data we found that the syntactic information contains clues about the differences between a Binding event with multiple Themes and multiple Binding events with one Theme.

Figure 2 exemplifies cases where a Binding event has multiple Themes. In these cases, the common path to the root of the dependency tree is the same for



(a) Multiple Themes for a Binding event



(b) Multiple Binding events

Figure 2: Multiple Themes for a Binding event versus multiple Binding events

all Themes and for the event trigger. In Figure 2a the common path between the dashed lines (the Themes) is also common with the doubled line (the event trigger). If there are multiple Binding events –all with one Theme– the common path of the Themes is longer than the part that they share with the event trigger, which might indicate that there is a syntactic construction that acts as a coordination. Figure 2b shows that the common path for the dashed lines contains the token "with". The path to this token is not common with the path of the event trigger. The system uses this path information to disambiguate between multiple Themes for a Binding event and multiple Binding events.

5.3 The nested events rule

Another rule takes care of nested events, like the Regulation event in Table 6.8. For the Gene Expression event in Table 6.8 the first rule created four events from the same event trigger. The rule for nested events will add as many Regulation events as there are new Theme events, in this case Gene Expression events. As a result of the first rule and this rule, the system outputs eight events: four Gene Expression events and four Regulation events, which, in this case, happens to be the same as the gold standard.

6 Results and discussion

The results that we report in this Section have been obtained by training the system on the training corpus and testing it on the test corpus via the web service provided in the web page of the BioNLP Shared Task[3]. The final results are presented in Table 6.9.

	Total	Precision	Recall	F1
Binding	347	43.46	23.92	30.86
Gene Expression	722	74.12	52.35	61.36
Localization	174	76.74	37.93	50.77
Phosphorylation	135	69.05	64.44	66.67
Protein Catabolism	14	30.00	21.43	25.00
Transcription	137	60.26	34.31	43.72
Regulation	291	33.94	12.71	18.50
Positive Regulation	983	36.59	19.43	25.38
Negative Regulation	379	39.33	15.57	22.31
TOTAL	3182	53.37	29.89	38.32

Table 6.9: Results of the system (ASM/ARM) on the test set.

The scores show that the system does not process Regulation and Binding events at a satisfactory competitive level, like most participating systems[4]. Both, precision and recall are low. Protein Catabolism also gets low scores, but the frequency of these events is extremely low, so the score is not reliable. Precision is acceptable for Gene Expression, Localization, Phosphorylation and Transcription events, and recall for Phosphorylation events. The system reaches an F1 score of 38.32 % in the ASM/ARM mode. The results of this system can be compared to the results of the systems that participated in the BioNLP-ST. There was a large range of variation in the results of participating systems (between 16 and 52% F1

[3]Web page of the evaluation server: `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/eval-test.shtml`.

[4]Results of all the participating systems can be found at `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/results/results-master.html`.

score). Compared to the best three systems of the task, the system presented in this paper scores 13.63, 8.34 and 6.3 % lower.

The three best systems used diverse approaches. In Björne et al. (2009), a graph representation and transformation approach is used in which the different steps of the processing (trigger detection and role detection) are graph transformations (adding or removing nodes and edges) achieved using a multi-class SVM. A hand-crafted rule-based postprocessing step patches the output (e.g. removing extra themes for events that can have only one theme). In Buyko et al. (2009), manually curated dictionaries were used as extra information for event trigger identification. This approach starts from dependency parses which are simplified and decorated with conceptual class information. Maximum entropy and graph kernel SVM machine learning algorithms were used for classification (sometimes combined in an ensemble) in different combinations for different types of events. A postprocessing module is used here as well. In Kilicoglu and Bergler (2009), a rule-based system was used, operating on dependency parses and using manually curated dictionaries like UMLS for trigger identification.

Compared to a memory-based system (Morante et al. 2009) that participated in the BioNLP-ST, this system scores 7.75 % F1 higher. The system described in (Morante et al. 2009) solves the classification task in two not-joint learning steps, as most systems do. First, a token-based classification finds event triggers, and then a pairwise classification finds participants. Additionally, that system uses more rules for postprocessing, which are more complex. The difference in precision is 5.67 %, and in recall 7.39 %. The comparison would suggest that the joint approach combines better with memory-based learning, though this conclusion should be further explored, since the classifiers used in both systems do not use the same features. The main difference lies in the fact that the system presented here uses features from the dependency tree, whereas the system in Morante et al. (2009) uses mostly features from the sequential context in the sentence.

It is difficult to gain insight into the decisions made by the machine learner in order to explain the cause of misclassification errors. A low percentage of errors is caused by the lack of intersentential event finding, which amount to 5 % of the cases in the training data. Another percentage of errors is caused by the conversion from BioNLP-ST to CoNLL format (7.43% F1 on development data). Some errors are related to Binding events that have multiple Themes, and to Regulation events with nested structures. Another source of errors are overlapping events, that is, events that are triggered by the same token. Finally, the system has problems with detecting multitoken event triggers like “had only a slight effect”, which is a Positive Regulation event. In sum, apart from the difficulties of the classification tasks, a proportion of errors is caused by the conversion of the data into a learnable format and by the reconstruction of the output of the classification phase into event frames.

7 Conclusions

In this paper we presented a supervised machine learning system that extracts events from biomedical texts according to the definition of the BioNLP Shared Task 2009 (Kim et al. 2009). The main characteristic of the system is that it integrates classifiers that learn event triggers and event participants jointly, avoiding the traditional pipeline architecture prone to error propagation. Additional characteristics are that the system does not make use of external resources like ontologies or dictionaries, and that the rules needed to reformat the data into the original data files are kept to the minimum number making the system adaptable to any domain. The results show that this system is more efficient than a similar memory-based system that used shallow context information and integrated classifiers in a traditional pipeline architecture.

However, the fact that the system scores 13.63 % lower than the best system for the same task suggests that research has to continue in order to reach a more satisfactory performance. Further research will experiment with other algorithms and ensembles because we believe that, given the complexity of the task, taking profit of the positive aspects of different algorithms might increase the performance and help produce results at the level of other well established tasks. Additionally, research should focus on determining what type of domain knowledge would be useful to solve the task and on integrating it in the system.

Acknowledgments

This work was made possible through financial support from the University of Antwerp (GOA project BIOGRAPH).

References

Ananiadou, S. and J. McNaught (2006), *Text Mining for Biology and Biomedicine*, Artech House Books, London.

Björne, J., J. Heimonen, F. Ginter, A. Airola, T. Pahikkala, and T. Salakoski (2009), Extracting complex biological events with rich graph-based feature sets, *Proceedings of BioNLP 2009: Shared Task on Event Extraction*.

Buchholz, S. and E. Marsi (2006), CoNLL-X Shared Task on Multilingual Dependency Parsing, *Proceedings of X CoNLL: Shared Task*, New York.

Bundschus, M., M. Dejori, M. Stetter, V. Tresp, and H-P Kriegel (2008), Extraction of semantic biomedical relations from text using conditional random fields, *BMC Bioinformatics* **9**, pp. 207.

Buyko, E., E. Faessler, J. Wermter, and U. Hahn (2009), Event extraction from trimmed dependency graphs, *Proceedings of BioNLP 2009: Shared Task on Event Extraction*.

Chou, W.C., R.T.H. Tsai, Y-S. Su, W. Ku, T-Y Sung, and W-L Hsu (2006), A Semi-Automatic Method for Annotating a Biomedical Proposition Bank, *Proceedings of the ACL Workshop on Frontiers in Linguistically Annotated Corpora 2006*.

Daelemans, W. and A. van den Bosch (2005), *Memory-based language processing*, CUP.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch (2007), TiMBL: Tilburg Memory Based Learner, version 6.1, Reference Guide, *TR 07-07*, ILK, Tilburg, The Netherlands.

Finkel, J.R. and C.D. Manning (2009), Joint parsing and named entity recognition, *Proceedings of HLT-NAACL 2009*, Boulder, Colorado.

Kilicoglu, H. and S. Bergler (2009), Syntactic dependency based heuristics for biological event extraction, *Proceedings of BioNLP 2009: Shared Task on Event Extraction*, Boulder, USA.

Kim, J.D., T. Ohta, and J. Tsujii (2008), Corpus annotation for mining biomedical events from literature, *BMC Bioinformatics* **9**, pp. 10.

Kim, J.D., T. Ohta, and J. Tsujii (2009), Overview of BioNLP'09 Shared Task on Event Extraction, *Proceedings of BioNLP 2009: Shared Task on Event Extraction*, Boulder, USA.

Krallinger, M., A. Valencia, and L. Hirschman (2008a), Linking genes to literature: text mining, information extraction, and retrieval applications for biology, *Genome Biology* **9(Suppl 2)**, pp. S8.

Krallinger, M. and A. Valencia (2005), Text-mining and information-retrieval services for molecular biology, *Genome Biology* **6**, pp. 224.

Krallinger, M., F. Leitner, C. Rodriguez-Penagos, and A. Valencia (2008b), Overview of the protein–protein interaction annotation extraction task of BioCreative II, *Genome Biology* **9(Suppl 2)**, pp. S4.

Morante, R., V. Van Asch, and W. Daelemans (2009), A memory-based approach to event extraction in biomedical texts, *Proceedings of BioNLP 2009: Shared Task on Event Extraction*, Boulder, USA.

Nédellec, C. (2005), Learning Language in Logic – Genic Interaction Extraction Challenge, *Proceedings of LLL 2005*, Bonn, Germany.

Pyysalo, S., F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski (2007), BioInfer: a corpus for IE in the biomedical domain, *BMC Bioinformatics*.

Sagae, K. and J. Tsujii (2007), Dependency parsing and domain adaptation with LR models and parser ensembles, *Proceedings of CoNLL 2007: Shared Task*, Prague, Czech Republic, pp. 82–94.

Sasaki, Y., P. Thompson, P. Cotter, J. McNaught, and S. Ananiadou (2008), Event frame extraction based on a gene regulation corpus, *Proceedings of Coling 2008*, Manchester, UK.

Wang, X., J. Nie, D. Luo, and X. Wu (2008), A Joint Segmenting and Labeling Approach for Chinese Lexical Analysis, *Proceedings of the European conference on Machine Learning and Knowledge Discovery*, Springer Verlag, Berlin, pp. 538–549.

Wattarujeekrit, T., P.K. Shah, and N. Collier (2004), PASBio: predicate-argument structures for event extraction in molecular biology, *BMC Bioinformatics* **5**, pp. 155.

Comparison of applying Pair HMMs and DBN models in Transliteration Identification

Peter Nabende

Alfa-Informatica, University of Groningen
p.nabende@rug.nl

Abstract

Transliteration is aimed at dealing with unknown words in Cross Language Information Retrieval (CLIR) and Machine Translation (MT). Most of the transliteration tasks depend on a similarity estimation stage where a model is utilized with the aim of identifying a transliteration match for a given source word. In this paper, we evaluate the application of two related frameworks to transliteration identification. Both frameworks model string similarity as the cost incurred through a series of edit operations. One framework implements Pair Hidden Markov Models (Pair HMMs) (Mackay and Kondrak 2005) while the other implements classes of Dynamic Bayesian Network (DBN) models (Filali and Bilmes 2005). For each Pair HMM, we adapt different algorithms for computing transliteration similarity estimates. For the DBN framework, we modify the DBN classes in (Filali and Bilmes 2005) and specify models from the classes to represent factorizations that we hypothesize could affect the value of a transliteration similarity estimate. Separate tests applying models from the two frameworks result in high transliteration identification accuracy on an experimental setup of Russian-English transliteration. A check on the output from models associated with the two frameworks suggests that there can be improved transliteration identification accuracy through a combination of models.

1 Introduction

Transliteration tasks require analyzing strings where each of the languages uses a different writing system, for example determining the level of similarity between a string written in English "Czeladź" and its Russian representation "Челядзь". The main aim of a transliteration analysis process is to determine a correct representation of a string in a different writing system that is expected to bear a pronunciation similar to that of the original source word. It is no surprise that most of the earliest attempts at automated transliteration proposed phoneme-based approaches (Knight and Graehl 1998, Jung et al. 2000). Recently, approaches that consider only orthographic representations have resulted in comparable if not better machine transliteration performance than phoneme-based approaches (Li et al. 2004). Currently, various approaches are being sought to develop automated transliteration systems; the shared tasks on machine transliteration and transliteration mining (Li et al. 2009, Kumaran et al. 2010) represent such ongoing attempts at evaluating state of the art machine transliteration systems. In this paper, we investigate the use of two methods in automated transliteration identification: Pair HMMs and DBNs.

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

The two methods model transliteration as a cost incurred in transforming a source word to a target word through a series of edit operations. The three edit operations are: substitution, insertion, and deletion. Each of the methods utilizes different algorithms for estimating the edit cost as a similarity score from the edit based transformation of a source string to a target string. The edit cost is in turn used to find the most likely transliteration out of a set of candidate transliterations. The Pair HMM method specifies the edit operations as states in which we estimate parameters associated with source-target character relationsips including those with empty symbols. The DBN framework specifies source-target string transformation through factorizations on edit operations. The two methods have been proposed in previous related work (Filali and Bilmes 2005, Mackay and Kondrak 2005), and although there are comparisons when applied in the task of cognate identification (Kondrak and Sherif 2006), the proposal to apply the methods to transliteration tasks, also necessitates comparison. In this paper, we specify and test additional variants of models from both frameworks and later compare their performance. The models from both frameworks are also evaluated against those of a baseline method of using paired trigram statistics in the given source and target language corpora. We also check the transliterations identified from each model with the aim of determining whether there could be benefit in combining two or more models from each framework or from both frameworks for estimating transliteration similarity. The paper is organized as follows: section 2 describes the Pair HMM and DBN frameworks, section 3 suggests some of the challenges associated with using the two frameworks on data associated with different writing systems, section 4 describes the transliteration experimental setup and discusses results from testing different pair HMM and DBN models. Section 5 concludes the paper with pointers to future work.

2 Models for Transliteration similarity estimation

Transliteration similarity estimation can be looked at as determining the level of relationship between two strings in different writing systems. Different methods can be used in the process of measuring string similarity. A common approach that is followed in this paper uses the notion of edit distance where string similarity is associated with the cost of 'edit operations' required to transform one string to another string. Figure 1 illustrates the sequence of edit operations required to transform the Russian string "пётр" to the English string "Peter" (a) and the Dutch string "Pieter" (b). In Figure 1, the edit operations are denoted by M(Substitution), and I(Insertion). For the case where we would have aligned a character in the Russian string to a gap in the English or Dutch string, the edit operation is denoted as a D(Deletion). The Pair HMM and DBN methods represent this process differently, and we distinguish between them in the following subsections.

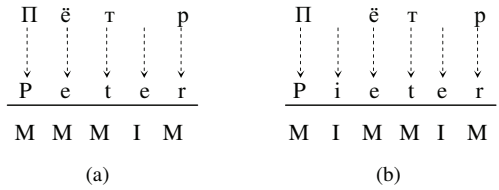


Figure 1: Illustration of alignment through edit operations required to transform a name written in Russian "пётр" using the Cyrillic alphabet with corresponding English (a) and Dutch (b) representations written using the Latin alphabet.

2.1 Edit distance based Pair HMM method

In a Pair HMM, edit operations such as those illustrated in Figure 1 are defined as "emission" states that are used to model the relationship between source and target language characters. The probabilistic relationship for each source language character and target language character is modeled in the substitution state (M); that for each source language character and an empty target language symbol is modeled in the deletion state (D); and that for each each target language character and source language empty symbol is modeled in the insertion state (I). We also have the option to represent transitions between a Pair HMM's states in various ways. A transition parameter represents the probabilistic value associated with moving from one state to another or the same state for a given Pair HMM. While using Pair HMMs, the main focus is usually on comparing the effectiveness of different Pair HMM algorithms and determining the optimal structure of the underlying model. Generally, to determine the optimal Pair HMM structure, we can examine the relative contribution of three sets of parameters (Mackay and Kondrak 2005): substitution parameters, gap parameters (insertion and deletion), and transition parameters. Because substitution parameters constitute the core of a Pair HMM, focus is usually put on the gap and transition parameters. In this paper, we determine the effect of Pair HMM transition parameters on the task of transliteration similarity estimation for a given language pair dataset. We have therefore specified four Pair HMM variants where we vary the size and properties of transition parameters.

The first Pair HMM variant does not use transition parameters between each of the edit states; it only uses transition parameters from a start state to one of the edit states, and from one of the edit states to the End state. The second Pair HMM variant (Figure 2(a)) uses three transition parameters (α, β, and δ), where each transition parameter is associated with leaving one of the edit states and the starting parameters are associated with the transition parameters from the substitution state to one of the edit states. The third Pair HMM variant (Figure 2(b)) is adapted from previous work (Mackay and Kondrak 2005, Wieling et al. 2007) where the starting parameters are also associated with the transition parameters from the substitution state. The last Pair HMM variant (Figure 3) uses nine distinct transition

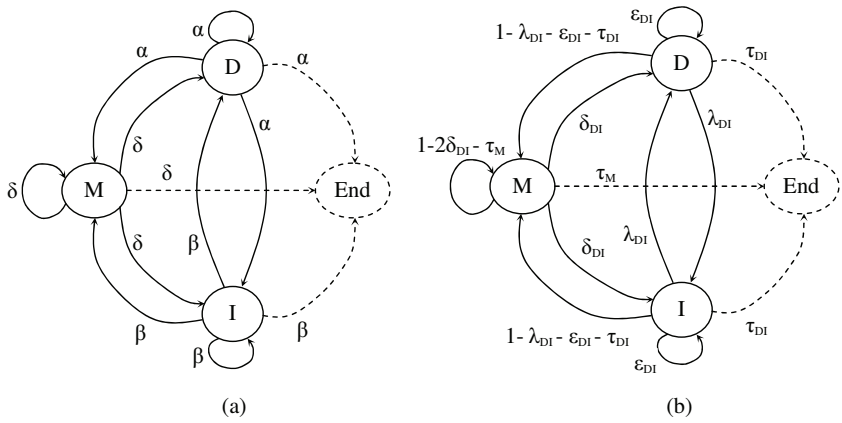


Figure 2: (a) Pair HMM with three transition parameters (b) Pair HMM with five transition parameters (Mackay and Kondrak 2005). Solid edges are associated with transitions to emitting states and solid nodes are emitting states. dotted edges transit to the non-emitting End state.

parameters between each of the Pair HMM states and the starting parameters have similar properties as those of the two previous Pair HMM variants. Each variant is designed to model the parameters in the insertion and deletion states distinctly.

For each Pair HMM variant, an implementation of the Pair HMM's Baum-Welch algorithm is used to estimate all its transition and emission parameters given source target language training data. Four different algorithms are defined for each Pair HMM variant and are used to compute similarity scores for source and target language strings given the corresponding Pair HMM. The scoring algorithms are (Mackay and Kondrak 2005): forward, Viterbi, Viterbi log odds, and forward log odds. For a brief description of these algorithms: the forward algorithm considers all possible alignments when determining a string similarity estimate; the Viterbi algorithm considers the best alignment(s); and the log odds versions normalize the base algorithm (forward or Viterbi) scores using a random model score. The random model assumes no relationship between sequences and captures the probability of a pair of symbols co-occuring by chance.

2.2 DBN-based edit distance method

The DBN-based edit distance method uses the graphical models approach of Dynamic Bayesian Networks where random variables are used to represent hidden edit operation states that are used in estimating string similarity. Generally, DBNs are used to represent both time-series data that is generated by some causal process and sequence (e.g. Natural Language Processing or Biological) data where

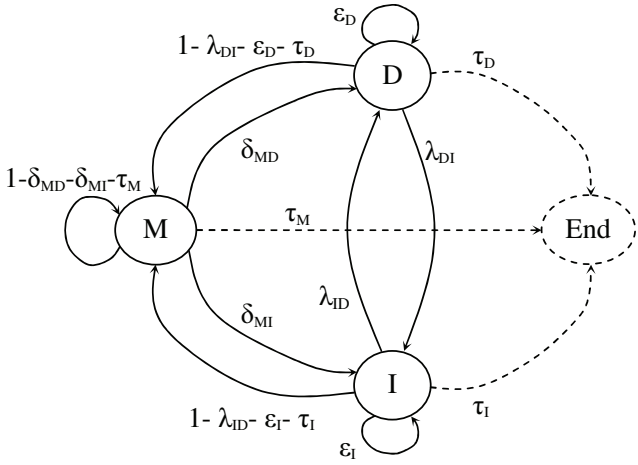


Figure 3: Pair HMM with nine transition parameters.

we are doubtful about the generating mechanism (Murphy 2002). DBNs generalize HMMs by representing the hidden state as an arbitrary set of random variables with arbitrary conditional independence assumptions (Zweig and Russell 1998).

The DBN method is implemented using the Graphical Modelling ToolKit (GMTK) (Bilmes and Zweig 2002) that simplifies the representation and modification of various types of models including HMMs. We have various options to vary the type of dependencies on the hidden edit operation states representing various factorizations that we postulate could affect the probabilistic similarity value associated with a pair of strings. Most of the models that were initially specified in (Filali and Bilmes 2005) are adapted in this paper. We start by adapting the base DBN model that is also referred to as the Memoryless and Context Independent (MCI) model. Figure 4 is shows the graphical templates for the start Bayesian Network (BN), *inter-slice* BN, chunk BN, and end BN for the MCI DBN model.

To help understand the graphical representations of the edit distance based DBN models, let us first review the stochastic extension of string edit distance in (Filali and Bilmes 2005) upon which the edit distance based DBN framework was developed. Given a source string $s_1^m = s_1 s_2 ... s_m$ of length *m* over an alphabet A_s, and a target string t_1^n of length *n* over an alphabet A_t; we can model edit operations using a hidden random variable Z, that takes values in $(A_s \cup \epsilon \times A_t \cup \epsilon) \setminus (\epsilon, \epsilon)$ where ϵ represents an empty string, and *Z* is perceived as a random vector with two components ($Z^{(s)}$ and $Z^{(t)}$). To estimate string similarity, we follow a similar approach, where we determine the joint probability $P(s_1^m, t_1^n|\theta)$ of observing the source/target string pair (s_1^m, t_1^n) given model parameters θ. In (Filali and Bilmes 2005), the probability of a particular pair of strings is expressed as the sum of the probabilities of all possible ways of generating the pair:

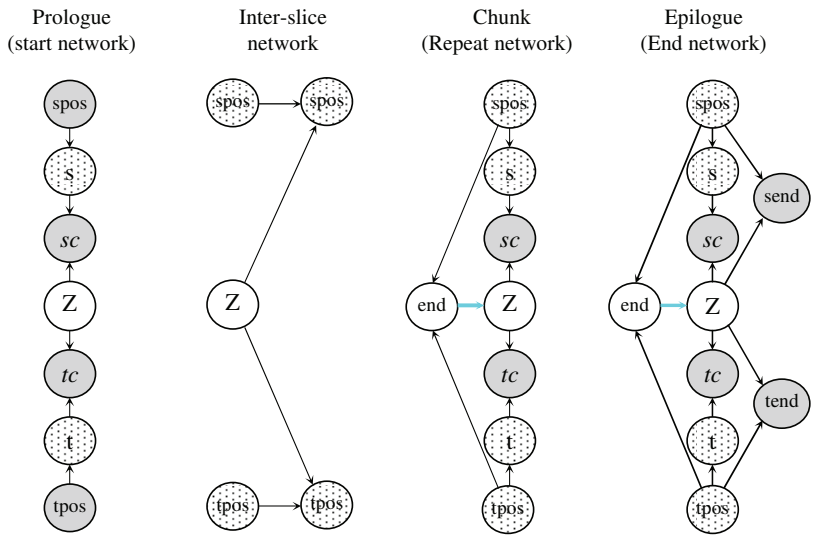


Figure 4: Graphical template for the MCI DBN model. Following the common convention for representing graphical models, shaded nodes represent observed variables, unshaded nodes represent hidden nodes, and nodes with dots represent deterministic hidden variables. Adapted from (Filali and Bilmes 2005).

$$P(s_1^m, t_1^n|\theta) = \sum_{z_1^l : v(z_1^l)=\langle s_1^m, t_1^n\rangle, max(m,n)\leq l\leq m+n} \sum P(z_1^l, s_1^m, t_1^n|\theta)$$

where $v(z_1^l)$ represents the string pair generated from the sequence z_1^l.

For the MCI DBN model, because there is no dependence between edit operations, $P(z_1^l, s_1^m, t_1^n|\theta)$ can be factored as $\prod_i P(z_i, s_1^m, t_1^n)$ where $1 < i < l$ and $z_i = \langle z_i^{(s)}, z_i^{(t)}\rangle$. It is also noted that although the term *context independent* is used for the MCI DBN model, there is a global dependence between z_i and source target string symbols that forces z_1^l to generate (s_1^m, t_1^n).

In Figure 4, Z represents the current edit operation variable. *spos* and *tpos* are variables representing the current position in the source and target strings respectively. *s* and *t* represent the current character in the source and target strings respectively. The *sc* and *tc* nodes are source and target consistency[1] nodes. The end node is a switching parent of Z and represents the variable that indicates when

[1]The *sc* and *tc* nodes have a fixed observed value 1 and the only configuration of their parents are such that the source component of the edit operation variable Z is *s* or an empty symbol for *sc* and t or an empty symbol for *tc* and that Z does not generate empty source and target symbols at the same time (Filali and Bilmes 2005)

we are past the end of both the source and target strings, i.e. when $spos > m$ and $tpos > n$. The *send* and *tend* nodes represent variables that ensure that we are past the end of the source and target strings respectively.

Based on the MCI model, we have currently adapted three other DBN model classes that were initially introduced in (Filali and Bilmes 2005). The DBN model classes represent different dependencies on the edit operation random variables and include: edit operation memory dependencies; source and / or target character context dependencies; and edit operation length dependencies. We briefly point out the main properties of these other DBN models.

Figure 5 represents the interslice network for a context independent Memory (MEM) DBN model. The starting, chunk, and end networks for the MEM DBN model are not included in Figure 5 since they are similar to those of the MCI DBN model (Figure 4). In Figure 5, a variable H that can be used to implement various dependencies with Z_i is introduced. Generally, H can be stochastic or deterministic, and its cardinality determines the amount of information that can be "summarized" from one slice to another (Filali and Bilmes 2005). In this paper,

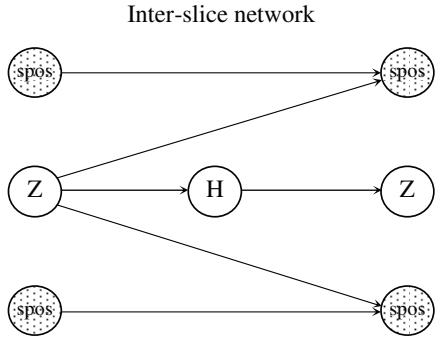


Figure 5: Inter-slice graphical template for the context independent Memory model. Adapted from (Filali and Bilmes 2005).

we only test the deterministic implementation of H, where H is a simple copy of Z, which is as well represented by the specification $P(Z_i|Z_{i-1})$.

Figure 6 represents the starting, inter-slice, chunk, and end networks for a Context (CON) dependent DBN model, where we add edges from s_i, $sprev_i$ to Z_i. In the other CON DBN model that is tested, we only add an edge from s_i to Z_i.

Figure 7 shows the graphical templates for the inter-slice and intra-slice networks for the MCI length DBN model. Additional variables are used to specify the logic needed to factor in the length of the edit sequence in the final transliteration similarity estimate. In Figure 7, *incl*, represents a random variable that determines the number of allowed edit operations. The variable *cnt* is used to determine the slice number and is used to trigger the random variable *reql* when the required frame number is reached.

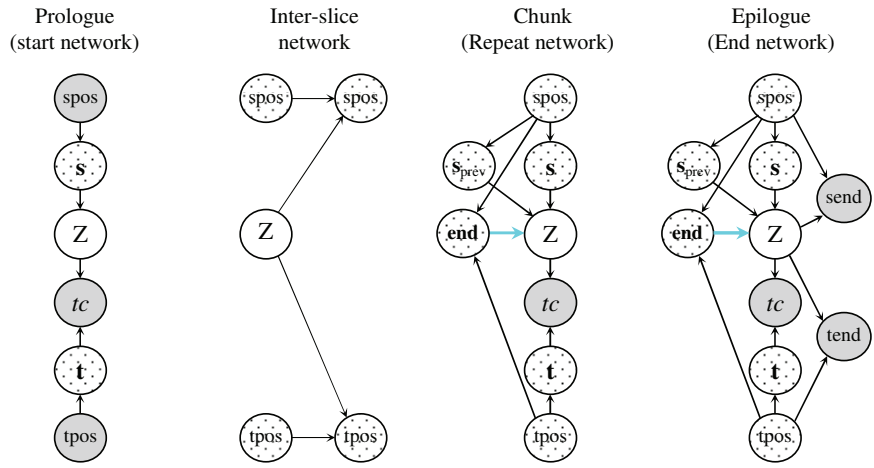


Figure 6: Graphical template for a context dependent DBN model.

We mainly modify the DBN models considering the type of data on which they are to be applied. For example in previous work, the DBN models were applied on data where the source and target language used the same writing system and hence the size and alphabets that were specified for both languages was the same. For the transliteration task, the source and target languages use many different characters and the total number of characters used for each writing system is not the same; we therefore modify the DBN models to represent the different characteristics inherent in the source-target language transliteration data.

For each DBN model, the training procedure requires and leads to the specification of various types of parameters including: a *triangulated* structure for the DBN model, conditional probability tables representing the dependencies between variables for both *inter-slice* and *intra-slice* networks for a given DBN; decision trees; deterministic relationships; etc. A generalized Expectation Maximization (EM) algorithm is used for each of the DBN models that are tested in this paper. Similar to the cases in (Filali and Bilmes 2005, Kondrak and Sherif 2006), we use only three iterations for the EM procedure to avoid over-fitting the models. We have observed this number of EM iterations to be optimal for transliteration data as well. To obtain similarity estimates for source target language strings, we use GMTK's junction tree algorithm which uses learned parameters from the EM step and also uses transliteration data dependent parameters.

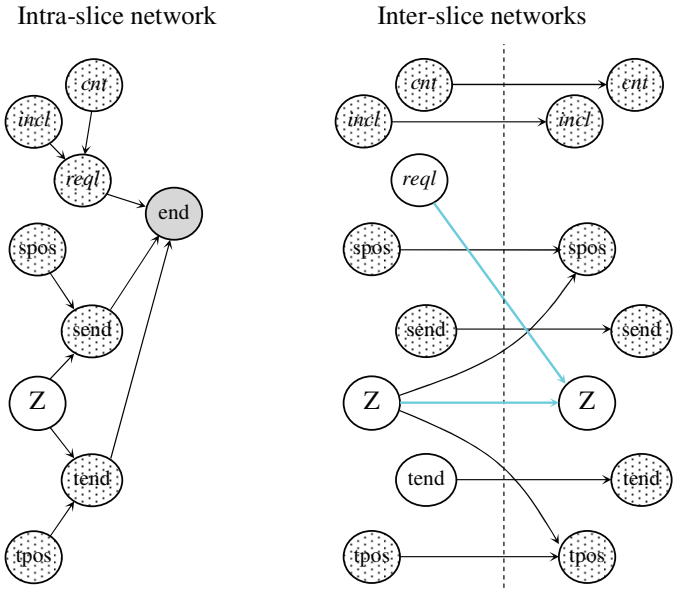


Figure 7: Graphical template for the MCI length DBN model. Some nodes and edges for example those associated with source and target symbols are omitted in this figure to simplify the description of additional variables for factoring in the edit sequence length. Adapted from (Filali and Bilmes 2005).

3 Challenges of applying edit distance based methods to transliteration

For the edit distance based methods, we specify tokens on single character basis. To improve processing, each token takes the form of a unique whole number. For example, if English is one of the languages and 26 characters are used in the data, each of the characters is converted to a whole number wn_i, where $0 \leq wn_i \leq 25$. This kind of approach to character representation is expected not to be suitable for all writing systems. Previous work (Filali and Bilmes 2005, Mackay and Kondrak 2005, Kondrak and Sherif 2006, Wieling et al. 2007, Nabende et al. 2010) suggests that tasks where languages use phonemic alphabets (for example the Latin alphabet) usually benefit from this approach. As a preliminary test, we applied the Pair HMM variant illustrated in Figure 3 on UTF-8 encoded Chinese-English transliteration data from the NEWS 2009 shared task on machine transliteration, and where Chinese is analyzed as the source language while English is the target language. In this case the writing system used for Chinese is mostly different from that of a phonemic alphabet and is mainly considered to be syllabic or logographic. For the Chinese-English transliteration data, 26 characters are used for the English part of the dataset and 370 simplified Chinese characters are used for the Chinese part of the dataset. We mapped each of the characters to unique whole numbers

in each of the respective languages and carried out an experimental transliteration identification run. We trained the Pair HMM variant on 31961 matching Chinese-English name pairs. We then tested the model on 2896 Chinese names in a held out test set. We obtained an accuracy of 0.213 and a Mean Reciprocal Rank (MRR) of 0.327 for the forward log odds algorithm which resulted in the best performance in this case. These preliminary results on the Chinese-English transliteration data suggest that the edit distance based methods that are tested in this paper are not valuable for all writing systems unless when a suitable representational approach is used. We intend to revisit this issue as future work where we propose to apply the Pair HMMs on a Romanized representation of the Chinese transliteration data.

4 Experiments

Two sets of experiments have been used: one set for Pair HMMs and the other for DBN models. The main difference in the two sets of experiments is the approach that is used to evaluate the models. A *stratified 10-fold cross validation* approach is used for the Pair HMMs while a *hold-out* method is used for the DBN models. The difference in testing approaches is mainly attributed to the amount of time it takes for some of the DBN models to execute to completion on the transliteration data-sets. The Pair HMMs are relatively faster on the same data-sets making it easy to follow the cross-validation approach. For both sets of experiments, UTF-8 encoded Russian-English datasets associated with the 2009 shared task on Machine Transliteration are used. When comparing models from each of the edit distance based framework, we use results from the respective sets of experiments. However, when comparing models across the two frameworks, only the *hold-out* method is used to enable evaluating more plausible DBN models.

4.1 Pair HMM Experiments

For the Pair HMMs, we split the English-Russian datasets into ten subsets that are mutually exclusive on the test set and where the set of characters used is the same as in the original dataset. Therefore, from a total of 7840 Russian-English transliteration pairs, we have 784 names in each subset as test data and the remaining data as training data. On the Russian side of the dataset, 34 characters constitute the alphabet while for English, 86 characters consitute the alphabet used for English. The large size of characters for English are mainly due to the large number of diacritics that are used for English in this particular dataset. We leave any diacritics in both datasets so as to maintain any associations that may exist regarding their rendering the other language.

The Cross Validation Accuracy (CVA) of any Pair HMM algorithm that is tested is simply calculated by averaging 10 individual accuracy measures:
$CVA = \frac{1}{10}\sum_{n=1}^{10} A_n$, where A_n is the accuracy of a model on the n^{th} test dataset,

$A_n = \frac{1}{N}\sum_{i=1}^{N} 1$ if $\exists r_{i,j} = c_{i,1}$; 0 otherwise, where $r_{i,j}$ is the j^{th} reference transliteration for the i^{th} name in the test set and $c_{i,1}$ is the first candidate transliteration that is identified based on a model's similarity estimate. N is the total number of names in each test set.

In a manner similar to that for CVA, the Cross Validation Mean Reciprocal Rank (CVMRR) is calculated by averaging the 10 individual MRR measures: $CVMRR = \frac{1}{10}\sum_{n=1}^{10} MRR_n$, where MRR_n is the MRR of a model on the n^{th} test set, $MRR_n = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{R(i)}$, where $R(i)$, is the rank of the correct matching transliteration candidate in a set of identified candidate transliterations associated with the i^{th} source language name in the test dataset. The values for CVA and CVMRR range from 0 to 1 and the closer the CVA or CVMRR is to 1, the better is a model's transliteration identification quality.

The CVA and CVMRR results for the different Pair HMM variant algorithms are shown in Table 7.1. The results suggest that it is important to model for Pair HMM transition parameters. The results show that transliteration identification accuracy improves when we increase the number of plausible transition parameters. However, there is only a slight improvement in accuracy for three algorithms of the Pair HMM variant with nine transition parameters (phmm_9_trans) compared to that with five transition parameters (phmm_5_trans). The results also suggest that the forward algorithm for each Pair HMM variant perform much better than the other algorithms. These transliteration identification results are different from those in (Mackay and Kondrak 2005) where the log odds algorithms performed best from the cognate identification task. As we reported from our preliminary test run on Russian-Chinese transliteration data, the results suggest that specific language pairs influence the relative performance of the Pair HMM algorithms.

4.2 DBN Experiments

For the DBN experiments, we use the first subset of data from the Russian-English transliteration data described in section 4.1. Excluding the division of the data into 10 subsets, the dataset is mantained as described in section 4.1. Since the *hold-out* method is used, the DBN models are evaluated using Accuracy and MRR for this particular dataset. In the next section, the results for the DBNs are presented against those for the Pair HMMs on the same dataset.

4.3 Results from applying DBN models and best Pair HMM algorithms

The evaluation results for the DBNs against the Pair HMMs are shown in Table 7.2. Table 7.2 also shows results from a *pair n-gram* (in this case *tri-gram*) that is used as a baseline. The *pair tri-gram* model uses counts based on the occurrence of source-target trigrams. The method works in such a way that the source and target

Pair HMM Variant	Scoring Algorithm	CVA	CVMRR
phmm_0_trans	Forward	0.9174	0.9473
	Viterbi	0.7958	0.8228
	Forward log odds	0.4189	0.5064
	Viterbi log odds	0.5296	0.6358
phmm_3_trans	Forward	0.9272	0.9522
	Viterbi	0.9222	0.9397
	Forward log odds	0.8906	0.9213
	Viterbi log odds	0.8003	0.8464
phmm_5_trans	Forward	**0.9806**	**0.9864**
	Viterbi	0.9302	0.9443
	Forward log odds	0.9159	0.9468
	Viterbi log odds	0.8343	0.8764
phmm_9_trans	Forward	**0.9798**	**0.9858**
	Viterbi	0.9342	0.9477
	Forward log odds	0.9208	0.9506
	Viterbi log odds	0.8451	0.8853

Table 7.1: CVA and CVMRR results for the Pair HMM variant algorithms. phmm_0_trans denotes the Pair HMM that uses no transition parameters; phmm_3_trans denotes the Pair HMM with three transition parameters (Figure 2(a)), phmm_5_trans and phmm_9_trans denote Pair HMMs with five transition parameters (Figure 2(b)) and nine transition parameters (Figure 3) respectively. The highest CVA and CVMRR values are in bold.

language strings are divided in overlapping *tri-grams* and the *tri-grams* are linked based on their position in the string. Related *tri-grams* are counted and the counts are used in estimating the similarity between two strings. In Table 7.2, we show results for the best performing Pair HMM algorithms on this dataset for each Pair HMM variant. phmm_0_trans, phmm_3_trans, phmm_5_trans, phmm_9_trans are as defined in Table 7.1 above. dbn_mci represents the base MCI DBN model (Figure 4); dbn_mci_len represents the MCI length DBN model (Figure 7); dbn_mem represents the MEM DBN model (Figure 5); dbn_con_s_i_s_{i-1} represents a CON DBN model where the edit operation random variable Z depends on the current (s_i) and previous (s_{i-1}) characters in the source string (Figure 6); dbn_con_s_i represents a CON DBN model where Z has a contextual dependency on the current character (s_i) in the source string; and dbn_con_s_i_len represents a CON length DBN model that factors in the dependency of Z on the current character (s_i) in the source string and the length of edit operations for computing the string similarity estimate.

Training times for the respective models are also included in the second column in Table 7.2. As shown in Table 7.2, the baseline model and Pair HMMs take shorter times for training, and the same is true for computing transliteration pair similarity values. Moreover, the training times for the Pair HMMs represent

Model	**Training Time (mins)**	**Accuracy**	**MRR**
baseline(pair trigram)	2.5	*0.9270*	*0.9450*
phmm_0_trans	< 3	0.9209	0.9501
phmm_3_trans	< 3	0.9401	0.9577
phmm_5_trans	< 3	0.9834	0.9898
phmm_9_trans	< 3	0.9834	0.9897
dbn_mci	146	0.9783	0.9853
dbn_mci_len	140	0.9031	0.9408
dbn_mem	34	0.8876	0.9210
dbn_con_s_i_s_{i-1}	49	0.9796	0.9834
dbn_con_s_i	22	**0.9872**	**0.9906**
dbn_con_s_i_len	21	0.9834	0.9887

Table 7.2: Accuracy and MRR Results from applying best Pair HMM variant algorithms and DBN models on 784 Russian-English test name pairs.

the total time over hundreds of iterations using the Buam-Welch algorithm on the Russian-English transliteration datasets. The DBNs on the other hand take relatively longer to train for the three specified EM iterations on the same transliteration dataset. It also takes longer while using DBN models in computing transliteration similarity estimates over the test dataset. However, the generic nature of the DBN framework guarantees a bigger model space for exploration, and there are already successful attempts at improving computational efficiency using DBNs.

For the qualitative measures, most of the edit distance based models perform better than the baseline method of pair trigrams apart from the dbn_mem and phmm_0_trans models. The results associated with the DBN models in Table 7.2 to a large extent are similar to those that were obtained from the pronunciation classification task in (Filali and Bilmes 2005). The results also show that the DBN context dependent models perform better than other DBN models. We also see that the context dependent DBN model (dbn_con_s_i) performs best overall although only slightly better than some of the best performing Pair HMMs and DBN models. The DBN MCI model, surprisingly performs better than some complex DBN models which is unlike the case in (Filali and Bilmes 2005). But this could be becuase of the nature of the English-Russian datasets in which both languages use a phonemic alphabet and a one to one character mapping may be already sufficient to compare a pair of strings. The results in Table 7.2 also suggest that additional information in some cases slightly lowers the transliteration identification accuracy of the DBN models on this particular dataset: for example attempting to factor information about the size of edit operations in a similarity estimate slightly lowers the model's transliteration identification accuracy although to an insignificant level. But this is also the case with the Pair HMMs where we have significantly better performance with the Pair HMM base Forward and Viterbi

algorithms as compared to the log odds algorithms that involve using additional information from a random model. The context dependent DBN models generally perform better, underlining the need for contextual representation in transliteration similarity estimation.

We have also looked through the transliteration identification output from some of the models. Table 7.3 shows the case where two models: a context dependent length DBN model and a Pair HMM forward algorithm did not provide the correct transliteration match at the first rank. A check on the output of the Pair HMM algorithms showed that they were failing on the same source name(s) while that for DBNs showed that they were mostly failing on different source names. Comparing the output of some of the Pair HMMs and the DBNs showed that a Pair HMM failed on many if not completely different source names when compared with a DBN model as is the case in Table 7.3. If the two models being compared were combined, the results show that the combination would have resulted in 100% transliteration identification accuracy. This seems to suggest that combining models from the two edit distance based methods could significantly improve transliteration identification accuracy. However, the challenge remains in deciding on what similarity estimate to use knowing that it is not yet possible to evaluate the two methods at the stage when they only provide a similarity estimate.

Context dependent DBN model			**Pair HMM(Forward algorithm)**		
Russian	**English**	**Rank**	**Russian**	**English**	**Rank**
Торунь	Toruń	34	Бештау	Beshtau	770
Спика	Spica	2	Ицпапалотль	Itzpapalotl	2
Бюзум	Büsum	2	Вайльтинген	Weiltingen	4
Фанагория	Phanagoria	5	Кураш	Kurash	2
Млынары	Młynary	2	Бернтайленд	Burntisland	2
Биттерфельд	Bitterfeld	12	Иккермюнде	Ueckermünde	2
Раштатт	Rastatt	2	Пролетариат	Proletariat	2
Эльстерауэ	Elsteraue	3	Файтсбронн	Veitsbronn	2
Трапштадт	Trappstadt	2	Валлетта	Valletta	2
Давидсон	Davidson	2	Плеоназм	Pleonasm	7
Куйбышев	Kuybyshev	596	Секстант	Sextant	2
Эгвекинот	Egvekinot	294	Радельфинген	Radelfingen	4
Тепейоллотль	Tepeyollotl	2	Арзамас	Arzamas	2

Table 7.3: Examples of where two models associated with the two edit distance based methods lead to identification of correct transliterations at ranks other than first rank. For these two models the table shows that they fail on completely different observation sequences and if combined could significantly complement each other.

In Table 7.3, we also observe some level of consistency in failing to identify strings at the first rank. For example we see that in most cases, where the Russian soft sign "ь" appears after one of the Russian characters (for example л 'l'), the

correct match is in most cases returned beyond the 2^{nd} rank.

5 Conclusion and Future Work

We have evaluated Pair HMMs and DBN models on an experimental setup of a transliteration identification task. We have shown that in their status as reported in this paper, models from the two frameworks lead to high transliteration identification accuracy. But also by looking at the identification results for each of the models, we propose that designing language dependent post-processing procedures after applying the edit distance based models could improve transliteration identification accuracy. Results from applying models in both frameworks also suggest that a combination of models would also lead to improved transliteration identification accuracy.

As future work, based on the results, we propose an investigation into ways of combining models from the two edit distance based methods methods for transliteration mining and transliteration generation. We also plan to investigate how models from the two edit distance based methods could be applied to other writing systems (for example Chinese) so as to result in improved transliteration identification.

Acknowledgements

Peter Nabende is funded through a second NPT Uganda Project (NPT-UGA-238). Thanks to the anonymous reviewers for their valuable comments. Finally, thanks to Erik Tjong Kim Sang for proof reading this manuscript and providing additional comments to enable a much better presentation of the work.

References

Bilmes, J. and G. Zweig (2002), The graphical models toolkit: An open source system for speech and time-series processing, *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 3916–3919.

Filali, K. and J. Bilmes (2005), A dynamic bayesian framework to model context and memory in edit distance learning: An application to pronunciation classification, *Proceedings of the Association for Computational Linguistics (ACL)*, Ann-Arbor, Michigan, pp. 338–345.

Jung, S.Y., S.L. Hong, and E. Paek (2000), An english to korean transliteration model of extended markov window, *Proceedings of the 18th Conference on Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 383–389.

Knight, K. and J. Graehl (1998), Machine transliteration, *Computational Linguistics* **24**, pp. 599–612, Association for Computational Linguistics.

Kondrak, G. and T. Sherif (2006), Evaluation of several phonetic similarity algorithms on the task of cognate identification, *LD'06: Proceedings of the Workshop on Linguistic Distances*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 43–50.

Kumaran, A., M. Khapra, and H. Li (2010), Whitepaper on news 2010 shared task on transliteration mining, Contains information for use in the 2nd shared task on Transliteration Mining.

Li, H., M. Zhang, and J. Su (2004), A joint source-channel model for machine transliteration, *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, Barcelona, Spain, pp. 159–166.

Li, H., V. Pervouchine, and M. Zhang (2009), Report of news 2009 machine transliteration shared task, *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS,2009)*, Association for Computational Linguistics, Suntec, Singapore, pp. 1–18.

Mackay, W. and G. Kondrak (2005), Computing word similarity and identifying cognates with pair hidden markov models, *Proceedings of the ninth Conference on Computational Natural Language Learning (CoNLL 2005)*, Ann-Arbor, Michigan, pp. 40–47.

Murphy, K.P. (2002), Dynamic bayesian networks: Representation, inference, and learning.

Nabende, P., J. Tiedemann, and J. Nerbonne (2010), Pair hidden markov model for named entity matching, *in* Tarek, S., editor, *Innovations and Advances in Computer Sciences and Engineering*, Springer Netherlands, pp. 497–502.

Wieling, M., T. Leinonen, and J. Nerbonne (2007), Inducing sound segment differences using pair hidden markov models, *SigMorphon '07: Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 48–56.

Zweig, G. and S. Russell (1998), Speech recognition with dynamic bayesian networks, *Proc. AAAI-98*, AAAI Press, Madison, Wisconsin.

Dutch Dependency Parser Performance Across Domains

Barbara Plank and Gertjan van Noord

Computational Linguistics, Faculty of Arts, University of Groningen

Abstract

In the past decade several natural language parsing systems have emerged, which use different methods and formalisms. For instance, systems that employ a hand-crafted grammar with a statistical disambiguation component versus purely statistical data-driven systems. What they have in common is the lack of portability to new domains: their performance might decrease substantially as the distance between test and training domain increases. Yet, to which degree do they suffer from this problem, i.e. which kind of parsing system is more affected by domain shifts? To address this question, we evaluate the performance variation of two kinds of dependency parsing systems for Dutch (grammar-driven versus data-driven) across several domains. We examine (1) how parser performance correlates to simple statistical properties of the text and (2) how sensitive a given system is to the text domain. This will give us an estimate of which kind of system is more affected by domain shifts, and thus more in need for domain adaptation techniques. To this end, we extend the statistical measures used by Zhang and Wang (2009a) for English and propose a new simple measure to quantify domain sensitivity.

1 Introduction

Most modern Natural Language Processing (NLP) systems are subject to the lack of portability to new domains: there is a substantial drop in their performance when the system gets input from another text domain (Gildea 2001). This is the problem of *domain adaptation*. Although this problem exists ever since the emergence of supervised Machine Learning, it has started to get attention only in recent years.

Studies on *supervised domain adaptation* (where there are limited amounts of annotated resources in the new domain) have shown that straightforward baselines (e.g. models based on source only, target only, or the union of the data) achieve a relatively high performance level and are "surprisingly difficult to beat" (Daumé III 2007). In contrast, *semi-supervised adaptation* (i.e. no annotated resources in the new domain) is a much more realistic situation but is also considerably more difficult. Current studies on semi-supervised approaches show very mixed results. Dredze et al. (2007) report on "frustrating" results on the CoNLL 2007 semi-supervised adaptation task for dependency parsing, i.e. "no team was able to improve target domain performance substantially over a state-of-the-art baseline". On the other hand, there have been positive results as well. For instance, McClosky et al. (2006) improved a statistical parser by self-training. Structural Correspondence Learning (Blitzer et al. 2006) was effective for PoS tagging and Sentiment

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

Classification (Blitzer et al. 2006, Blitzer et al. 2007), while only modest gains were obtained for structured output tasks like parsing.

For parsing, most previous work on domain adaptation has focused on *data-driven* systems (Gildea 2001, McClosky et al. 2006, Dredze et al. 2007), i.e. systems employing (constituent or dependency based) treebank grammars. Only few studies examined the adaptation of *grammar-based* systems (Hara et al. 2005, Plank and van Noord 2008), i.e. systems employing a hand-crafted grammar with a statistical disambiguation component. This may be motivated by the fact that potential gains for this task are inherently bound by the underlying grammar. Yet, domain adaptation poses a challenge for both kinds of parsing systems. But to what extent do these different kinds of parsing systems suffer from the problem? To address this question, we examine two particular issues:[1]

(Q.1) How does parser performance for Dutch correlate to simple statistical measures of the text?

(Q.2) How sensitive is a given system to the text domain, i.e. which parsing system (hand-crafted versus purely statistical) is more affected by domain shifts, and thus more in need for adaptation techniques?

The remainder of this paper is structured as follows. Section 2 provides an overview of related work. Section 3 and 4 introduce the different parsing systems, datasets and the experimental setup. Next, Q.1 is addressed in Section 5. Section 6 focuses on Q.2 and proposes a new simple measure to quantify domain sensitivity. In Section 7, conclusions are drawn and directions for future work are presented.

2 Related Work

For the statistical measures of the text and their correlation to parsing accuracy (Q.1) we start here from work by Zhang and Wang (2009a), who examined several state-of-the-art parsing models for English (WSJ and Brown). They show that different parsing models (constituent, dependency and deep-grammar based system) correlate on different levels to the three statistical measures examined (average sentence length, unknown word ratio and unknown part-of-speech trigram ratio). Their work directly inspired us. A related study is Ravi et al. (2008), who built a regression model to predict parser accuracy for English constituent parsing.

Regarding our second question (Q.2), to the best of our knowledge, no study has yet addressed this issue. Most previous work has focused on a single parsing system in isolation (Gildea 2001, Hara et al. 2005, McClosky et al. 2006). Recently, there is an observable trend towards combining different parsing systems to exploit complementary strengths. For instance, Nivre and McDonald (2008) combine two data-driven systems to improve dependency accuracy. Similarly, two studies successfully combined grammar-based and data-driven systems: Sagae

[1] Preliminary results of these research questions have been reported in Plank (2010) and Plank and van Noord (2010).

et al. (2007) incorporate data-driven dependencies as soft-constraint in a HPSG-based system for parsing the Wallstreet Journal. In the same spirit (but the other direction), Zhang and Wang (2009b) use a deep-grammar based backbone to improve data-driven parsing accuracy. They incorporate features from the grammar-based backbone into the data-driven system to achieve better generalization across domains. However, one issue remains open: which kind of system (hand-crafted versus purely statistical) is more affected by the domain, and thus more sensitive to domain shifts? We present an empirical evaluation of different parsing systems for Dutch, and propose a new simple measure to quantify domain sensitivity.

3 Parsing Systems

The parsing systems used in this study are: a grammar-based system coupled with a statistical disambiguation system (Alpino) and two data-driven systems (MST and Malt), described in the sequel.

(1) *Alpino* (van Noord 2006) is a deep-grammar based parser for Dutch that produces dependency structures as output. The system consists of approximately 800 grammar rules in the tradition of HPSG, and a large hand-crafted lexicon, that together with a left-corner parser constitutes the parser component. For words that are not in the lexicon, the system applies a large variety of unknown word heuristics (van Noord 2006), which among others attempt to deal with number-like expressions, compounds and proper names. The second stage of Alpino is a statistical disambiguation component based on Maximum Entropy. Thus, training the parser requires estimating parameters for the disambiguation component.

(2) *MST Parser* (McDonald et al. 2005) is a data-driven graph-based dependency parser. The system couples a minimum spanning tree search procedure with a separate second stage classifier to label the dependency edges.

(3) *MALT Parser* (Nivre et al. 2007) is a data-driven transition-based dependency parser. Malt parser uses SVMs to learn a classifier that predicts the next parsing action. Training instances represent parser configurations and the label to predict determines the next parser action.

Both data-driven parsers (MST and Malt) are thus not specific for the Dutch language, however, they can be trained on a variety of languages given that the training corpus complies with the column-based format introduced in the 2006 CoNLL shared task (Buchholz and Marsi 2006). Additionally, both parsers implement projective and non-projective parsing algorithms, where the latter will be used in our experiments on the relatively free word order language Dutch. Despite that, we train the data-driven parsers using their default settings (e.g. first order features for MST, SVM with polynomial kernel for Malt).

4 Datasets and Experimental Setup

The source domain on which all parsers are trained is cdb, the newspaper part of the Alpino Treebank (van Noord 2006). For our cross-domain evaluation, we consider Wikipedia and the Dutch Parallel Corpus (DPC). All are described next.

Source Cdb is a collection of text fragments from 6 Dutch newspapers, which has been annotated according to the guidelines of CGN (Oostdijk 2000) and stored in XML format. It consists of 140,000 words (7,136 sentences; average sentence length of 19.7 words). It is the standard treebank used to train the disambiguation component of the Alpino parser. Note that cdb is a subset of the training corpus used in the CoNLL 2006 shared task (Buchholz and Marsi 2006). The CoNLL data additionally contained a mix of non-newspaper text (namely, a large amount of questions from CLEF, roughly 4k questions, and around 1.5k hand-crafted sentences used during the development of the grammar), which we exclude here on purpose to keep a clean baseline.

Target The Wikipedia and DPC subpart of the LASSY corpus[2] constitute our target domains. These corpora contain several subdomains, e.g. sports, locations, science, communication (in total 10 Wikipedia and 13 DPC subdomains). A detailed overview of the corpora is given in Table 8.1. Note that both consist of hand-corrected data labeled by Alpino. This might introduce a slight bias towards Alpino, however it has the advantage that all domains employ the same annotation scheme. This avoids the problem of having differences in annotation guidelines, which was the major source of error in the CoNLL 2007 shared task on domain adaptation (Dredze et al. 2007).

CoNLL2006 This is the test file for Dutch that has been used in the CoNLL 2006 shared task on multi-lingual dependency parsing. The file consists of 386 sentences from an institutional brochure (about 'Jeugdgezondheidszorg'/youth healthcare) with an average sentence length of 15.2 words. We will use this file to check our data-driven models against state-of-the-art performance (Section 5.1).

Alpino to CoNLL format In order to train the MST and Malt parser and evaluate it on the various Wikipedia and DPC articles, we needed to convert the Alpino Treebank format into the tabular CoNLL format. To this end, we adapted the treebank conversion software developed by Erwin Marsi for the CoNLL 2006 shared task on multi-lingual dependency parsing. Instead of using the PoS tagger and tagset used in the CoNLL shared task (to which we did not have access at the time of these experiments), we replaced the PoS tags with more fine-grained tags obtained by parsing the data with the Alpino parser.[3] At testing time, the data-driven parsers are given the PoS tagged data as input, while Alpino uses plain sentences.

Evaluation In all experiments, unless otherwise specified, performance is measured as Labeled Attachment Score (LAS), the percentage of tokens with the correct dependency edge and label. To compute LAS, we use the CoNLL 2007 eval-

[2]LASSY (Large Scale Syntactic Annotation of written Dutch), ongoing project. Corpus version 17905, obtained from `http://www.let.rug.nl/vannoord/Lassy/corpus/`

[3]As will be discussed later (Section 5.1, cf. Table 8.2), using Alpino tags actually improved the performance of the data-driven parsers significantly. We could perform this check as we recently got access to the tagger and tagset used in the CoNLL shared task (Mbt with wotan tagset; thanks to Erwin Marsi).

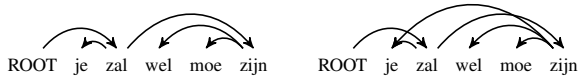
Wikipedia	**Wikipedia articles (excerpt)**	**#articles**	**#sentences**	**#words**	**ASL**
LOC (location)	België, Brussel (stad)	31	2190	25259	11.5
KUN (arts)	School van Tervuren	11	998	17073	17.1
POL (politics)	Belgische verkiezingen 2003	16	983	15107	15.4
SPO (sports)	Spa-Francorchamps, Kim Clijsters	9	877	9713	11.1
HIS (history)	Geschiedenis van België	3	468	8396	17.9
BUS (business)	Algemeen Belgisch Vakverbond	9	405	4440	11.0
NOB (nobility)	Albert II van België	6	277	4179	15.1
COM (comics)	Suske en Wiske	3	380	4000	10.5
MUS (music)	Sandra Kim, Urbanus (artiest)	3	89	1296	14.6
HOL (holidays)	Feest Vlaamse Gemeenschap	4	43	524	12.2
Total		**95**	**6710**	**89987**	**13.4**

DPC	**Description/Example articles**	**#articles**	**#sentences**	**#words**	**ASL**
Science	medicine (Zyprexa); Oceanography	69	3159	60787	19.2
Institutions	politics (Toespraak MP Kok)	21	1777	28646	16.1
Communication	ICT/Inet (DNS registreren)	29	1524	26640	17.5
Welfare state	pensions (Informatie over de AOW)	22	1130	20198	17.9
Culture	background articles (darwinisme)	11	791	16237	20.5
Economy	business texts (Inflatie op maat)	9	794	14722	18.5
Education	school (onderwijs in vlaanderen)	2	733	11980	16.3
Home affairs	presentation (Brussel, je hoofdstad)	1	540	9340	17.3
Foreign affairs	EU (Toespraak Cox EU raad)	7	372	9007	24.2
Environment	threat/nature(Koude oorlog noord-pool)	6	419	8534	20.4
Finance	banks (Opleiding private bankiers)	6	275	6127	22.3
Leisure	various (seks- en drugsschandaal)	2	140	2843	20.3
Consumption	toys (speelgoed uit China)	1	58	1310	22.6
Total		**186**	**11712**	**216371**	**18.5**

Table 8.1: Overview Wikipedia and DPC corpus. ASL = average sentence length.

uation script[4] with punctuation tokens excluded from scoring (as was the default setting in CoNLL 2006). We thus evaluate all parsers using the same evaluation metric. Note that the standard metric for Alpino is a variant of LAS, which allows for a discrepancy between expected and returned dependencies. Such a discrepancy can occur, for instance, because the syntactic annotation of Alpino allows words to be dependent on more than a single head (van Noord 2006). However, such 'secondary edges' are ignored in the CoNLL format; just a single head per token is allowed. The following example illustrates this:

CoNLL: *Alpino:*



Furthermore, there is another simplification. As the Dutch tagger used in the CoNLL 2006 shared task did not have the concept of multiwords, the organizers chose to treat them as a single token (Buchholz and Marsi 2006). We here follow the CoNLL 2006 task setup. Thus, to evaluate the Alpino output we convert it to CoNLL format using the same software.

[4] `http://nextens.uvt.nl/depparse-wiki/SoftwarePage`

5 Parser Performance and Simple Measures of the Text

We follow Zhang and Wang (2009a) and look at this stage at simple characteristics of the dataset without looking at syntactic annotation. We are interested in their correlation to parsing performance for Dutch.

Statistical measures We depart from the measures of Zhang and Wang (2009a) and add a perplexity measure estimated from a word-trigram Language Model. Thus the statistical measures used are:

Average Sentence Length (ASL) measures the average sentence length. Intuitively, longer sentences should be more difficult to parse than shorter ones.

Simple Unknown Word Rate (sUWR) calculates how many words (tokens) in the dataset have not been observed before, i.e. are not in the cdb corpus. For the Alpino parser, we use the percentage of words that are not in the lexicon (*aUWR, Alpino Unknown Word Rate*).

Unknown PoS Trigram Ration (UPTR) calculates the number of unknown PoS trigrams with respect to the original cdb training data.

Perplexity is the perplexity score assigned by a word-trigram language model estimated from the original cdb training data using the SRILM toolkit[5]. This feature, also used by Ravi et al. (2008), is intended as a refinement of UWR.

5.1 Empirical Results

Sanity checks First of all, we performed several sanity checks. We trained the MST parser on the entire original CoNLL training data as well as the cdb subpart only, and evaluated it on the original CoNLL test data. As shown in Table 8.2 (row 1-2) the accuracies of both models falls slightly below state-of-the-art performance (row 5), most probably due to the fact that we used standard parsing settings (e.g. no second-order features for MST). More importantly, there was basically no difference in performance when trained on the entire data or cdb only.

Model	**LAS**	**UAS**
MST (original CoNLL)	78.35	82.89
MST (original CoNLL, cdb subpart)	78.37	82.71
MST (cdb retagged with Alpino)	82.14	85.51
Malt (cdb retagged with Alpino)	80.64	82.66
MST (Nivre and McDonald 2008)	79.19	83.6
Malt (Nivre and McDonald 2008)	78.59	n/a
MST (cdb retagged with Mbt)	78.73	82.66

Table 8.2: Performance of the data-driven parsers versus state-of-the-art performance (McDonald et al. 2005; Nivre & McDonald, 2008) on the CoNLL 2006 test set (in Labeled/Unlabeled Attachment Score).

[5] http://www-speech.sri.com/projects/srilm/

We then trained the MST and Malt parser on the cdb corpus converted into the retagged CoNLL format, and tested on CoNLL 2006 test data (also retagged with Alpino). As seen in Table 8.2 (row 3 to 6), using Alpino tags improves the performance level significantly ($p < 0.002$, Approximate Randomization Test with 1000 iterations). This increase in performance can be attributed to two sources: (a) improvements in the Alpino treebank itself over the course of the years, and (b) the more fine-grained PoS tagset obtained by parsing the data with the deep grammar. To examine the contribution of each source, we trained an additional MST model on the cdb data but tagged with the same tagger as in the CoNLL shared task (Mbt, cf. Table 8.2 last row): the results show that the major source of improvement actually comes from using the more fine-grained Alpino tags ($78.73 \rightarrow 82.14 = +3.41$ LAS), rather than the changes in the treebank ($78.37 \rightarrow 78.73 = +0.36$ LAS). Despite the rather limited training data and use of standard training settings, we are in line with (and actually above) current results of data-driven parsing for Dutch.

We now turn to the various statistical measures. The parsers were all evaluated on the 95 Wikipedia articles.

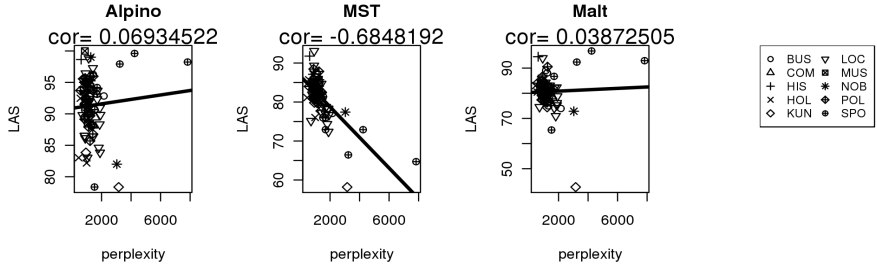


Figure 1: Pre-results (on 95 Wiki articles): Parser performance against sentence perplexity, including correlation coefficient – 3 peculiar sports articles fall out (crossed dots).

Pre-result While plotting the correlation between parser performance and statistical measure, three datasets immediately caught our eyes (the crossed dots; cf. Figure 1; we here only plot the perplexity graphs due to space reasons). These are three sports (SPO) articles about bike races. By inspecting them we notice that they contain a long list of winners from the various race years (on average 86% of the articles constitute this 'winner list'). Thus, despite the average short sentence length (6.03 words per sentence; in contrast to an average sentence length on Wikipedia of 13.4 words), the parsers exhibit very different performance levels on these datasets. Alpino, which includes various unknown word heuristics and a named entity tagger, is rather robust against the very high unknown word rate and reaches a very high accuracy level on these datasets. The Malt parser also reaches a high performance level on these special datasets. In contrast, the MST parser is

more influenced by unknown words, and its performance on these articles drops to its lowest level. These three sports articles thus form 'outliers' and we exclude them from the following experiment.

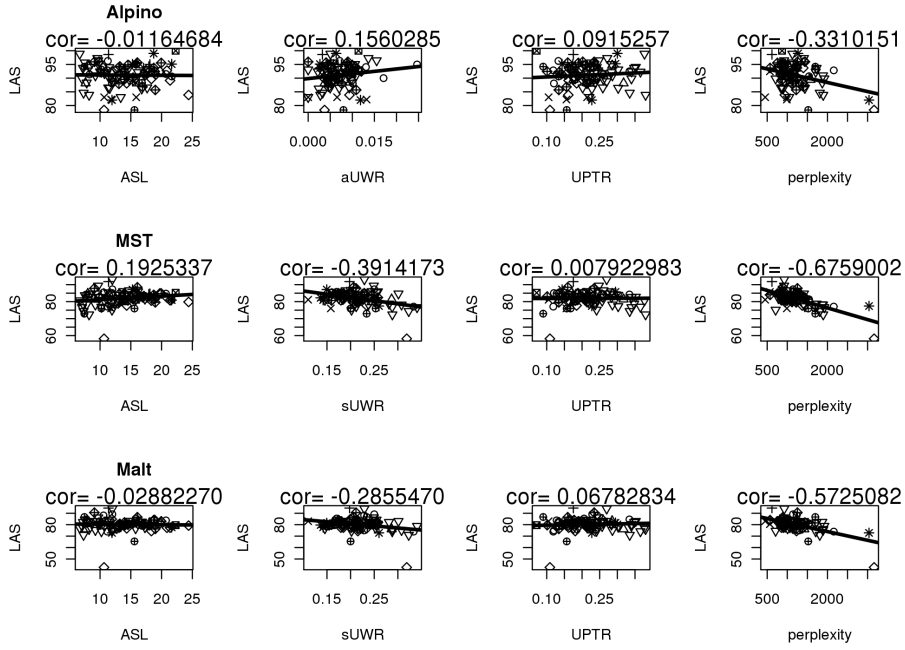


Figure 2: Results: Parser performance on a per-article basis against statistical measure on the text (93 Wikipedia articles - 3 sports articles removed).

Results Figure 2 depicts the correlation between parser performance and the four statistical measures of the text: Average Sentence Length (ASL), simple/Alpino Unknown Word Rate (sUWR/aUWR), Unknown PoS Trigram Rate (UPTR) and perplexity. All parsers are robust to Average Sentence Length (leftmost graphs in Figure 2). They basically do not show any correlation with this measure. This is in line with the results of Zhang and Wang (2009a) for MST and Malt. It is different for the grammar-based parsing system. Their grammar-based parser (ERG/PET) is highly sensitive to Average Sentence Length (correlation coefficient of -0.61 on their datasets), as longer sentences "lead to a sharp drop in parsing coverage of ERG" (Zhang and Wang 2009a). This is not the case for the Alpino parser. The system suffers less from coverage problems and is thus not so sensitive against increasing sentence length.

For Unknown Word Rate (UWR), the data-driven parsers show a high correlation (for this type of task) with this measure (correlation of -0.39 and -0.28),

which is in line with previous findings (Zhang and Wang 2009a). This is not the case for Alpino: again, its very good handling of unknown words makes the system robust to UWR. Note that for Alpino the unknown word rate is measured in a slightly different way (i.e. words not in the lexicon). However, if we would apply the same simple unknown word rate (sUWR) measure to Alpino, it would also result in a weak negative correlation only ($sUWR = -0.07$).

No parser does show any correlation with the third measure, Unknown Part-of-Speech Trigram Rate (UPTR). This is contrary to previous results (Zhang and Wang 2009a), most probably due to the usage of a different tagset and the freer word order language.

Our last measure, perplexity, exhibits the highest correlation to parsing performance: all parsers show the highest sensitivity against this measure, with the data-driven parsers being more sensitive ($cor = -0.67$ and -0.57) than the grammar-driven parser (-0.33). Note that this still holds if we would remove two other possible 'outliers', the diamond and star on the rightmost graphs of Figure 2, resulting in a correlation coefficient of -0.12 (Alpino), -0.57 (MST) and -0.34 (Malt). Moreover, also on DPC (as well as both together; graphs are omitted due to space limits) sentence perplexity gave us the highest correlation to parser performance.[6]

6 Sensitivity of Different Parsing Systems to the Text Domain

We now turn to the second question (Q.2). Clearly, the problem of domain dependence poses a challenge for both kinds of parsing systems, data-driven and grammar-driven. However, to what extent? Which kind of parsing system is more affected by domain shifts? We may rephrase our question as: Which parsing system is more robust to different input texts? To address this issue, we will examine the robustness of the different parsing systems in terms of variation of accuracy on a variety of domains. Note that the goal of this section is not so much to compare individual parser performances, but rather to examine the variability of parser performance across domains.

Towards a measure of domain sensitivity Given a parsing system (p) trained on some source domain and evaluated on a set of N available labeled target domains, the most intuitive measure would be to simply calculate mean (μ) and standard deviation (sd) of the performance on the target domains:

LAS_p^i = accuracy of parser p on target domain i

$$\mu_p^{target} = \frac{\sum_{i=1}^{N} LAS_p^i}{N} \qquad sd_p^{target} = \sqrt{\frac{\sum_{i=1}^{N}(LAS_p^i - \mu_p^{target})^2}{N-1}}$$

However, standard deviation is highly influenced by outliers. Furthermore, this measure does not take the source domain performance (baseline) into consideration nor the size of the target domain itself. We thus propose to measure the do-

[6] One could argue that the cdb corpus might be too small for perplexity scores; however, by using a much larger model estimated by adding the Twente Newspaper Corpus (500 million words) to cdb, the same conclusions are drawn. Perplexity remains the best statistical measure.

main sensitivity of a system, i.e. its *average domain variation* (adv) in accuracy, as weighted average difference from the baseline (source) mean, where the weights represent the size of the various domains:

$$adv = \frac{\sum_{i=1}^{N} w^i * \Delta_p^i}{\sum_{i=1}^{N} w^i} \quad \text{with } \Delta_p^i = LAS_p^i - LAS_p^{baseline},\ w^i = \frac{size(w^i)}{\sum_i^N size(w^i)}$$

In more detail, we propose to measure average domain variation relative to the baseline (source domain) performance by considering non-squared differences from the out-of-domain mean and weigh it by domain size. We thus want the adv measure to take on positive or negative values. Intuitively, to indicate the average weighted gain or loss in performance, relative to the source domain. We will examine this measure in the empirical result section to evaluate the domain sensitivity of the parsers, where $size$ will be measured in terms of number of words (given in Table 8.1). Furthermore, we will measure accuracy per subdomain, not on an article basis, to get more robust statistics.

Baselines To establish our baselines, we perform 5-fold cross validation for each parser on the source domain (cdb corpus, newspaper text). The baselines for each parser are given in Table 8.3.

Model	Alpino	MST	Malt
Baseline (LAS)	90.76	83.63	79.95
Baseline (UAS)	92.47	88.12	83.31

Table 8.3: Baseline (5-fold cross-validation). All differences are significant at $p < 0.001$.

Parser performance across domains As our goal is to assess performance variation across domains, we evaluate each parser on the Wikipedia and DPC corpora that cover a variety of domains (Table 8.1). Figure 3 and Figure 4 summarize the results for each corpus, respectively. In more detail, the figures depict for each parser the baseline performance as given in Table 8.3 (straight lines) and the performance on every domain (bars). Note that domains are ordered by size (number of words), with largest domains on the left. Bars filled with shading lines represent domains in which the parser achieves above-baseline performance, while full-colored bars indicate domains on which the parser's performance falls below source domain baseline, and applying domain adaptation techniques might be fruitful.

Figure 3 depicts parser performance on the Wikipedia domains with respect to the source domain baseline. The figure seems to suggest that the grammar-driven parser Alpino suffers the least from domain shifts. Besides the fact that Alpino scores high overall, its performance is above baseline on several Wikipedia domains. In contrast, the MST parser suffers the most from the domain changes; on most domains a substantial performance drop can be observed. The transition-based parser Malt scores on average lower than the graph-based counterpart, but is less affected by domain shifts than MST and thus lies somewhere in between.

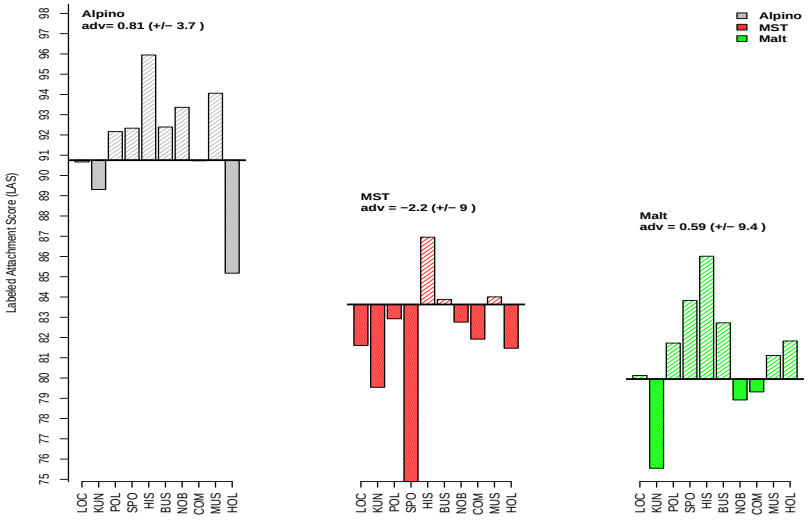


Figure 3: Performance on Wikipedia domains with respect to the source baseline (newspaper text) including average domain variation (adv) score. Domains are ordered by size (largest on left). Full-colored bars indicate domains where performance lies below baseline.

We can summarize these findings by using our proposed average domain variation measure: on average (over all Wikipedia domains), Alpino suffers the least ($adv = +0.81$) - it often scores above baseline, which our measure also suggests (positive score). Alpino is followed by Malt ($adv = +0.59$), also slightly gaining on some domains, and MST ($adv = -2.2$), which on average loses about 2.2 absolute LAS. Thus, MST is clearly the most domain sensitive parser, as also suggested in the graph by the many bars falling below baseline.

The results for the DPC corpus are depicted in Figure 4. It contains a broader set of domains, amongst others science texts (medical texts from the European Medicines Agency as well as texts about oceanography) and articles with more technical vocabulary (Communication, i.e. Internet/ICT texts). Both Malt and Alpino score above baseline on several domains, with this time presumably Malt being slightly less domain affected than Alpino (most probably because Malt scores above baseline on the more influential/larger domains). As with Wikipedia, the figure suggests also here that the MST parser is the most domain-sensitive parser. Our measure supports this finding: MST obtains a negative score ($adv = -0.27$), while Alpino ($adv = 0.22$) and Malt ($adv = 0.4$) achieve on average a gain over the baseline, with Malt being slightly less domain affected

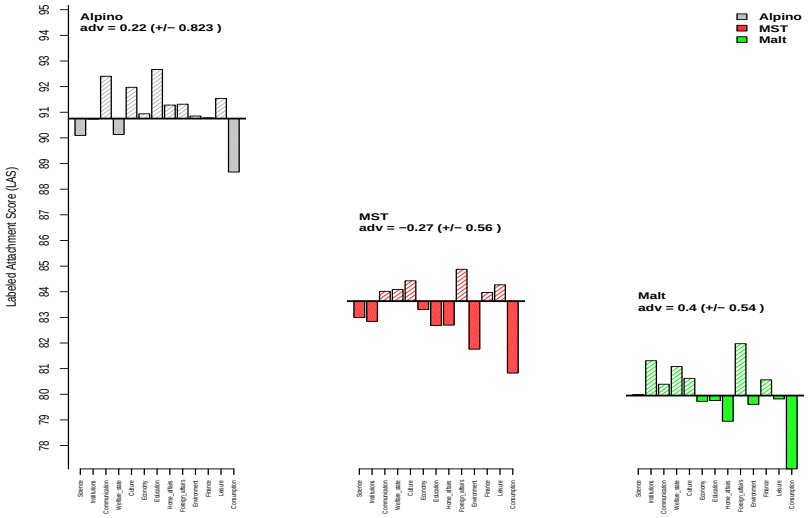


Figure 4: Performance on DPC domains with respect to the source baseline (cdb).

than Alpino. In contrast, if we would take only the deviation on the target domains into consideration (without considering the baseline and the domain size, as discussed in Section 6), we would get a completely opposite ranking on DPC: now the Malt parser would actually be considered the most domain-sensitive (here higher sd means higher sensitivity): Malt ($sd = 1.20$), MST ($sd = 1.14$), Alpino ($sd = 1.05$). However, by looking at Figure 4, intuitively, MST suffers more from the domain shifts than the other parsers, as most bars lie below the baseline. Moreover, the standard deviation measure neither gives a sense of whether the parser on average suffers a loss or gain over the new domains, nor incorporates the information of domain size. We thus believe that our proposed average domain variation is a better suited measure.

To check whether the differences in performance variation are statistically significant, we performed an Approximate Randomization Test over the performance differences (deltas) on the 23 domains (DPC and Wikipedia). The results show that the difference between Alpino and MST is significant. The same goes for the difference between MST and Malt. Thus Alpino is significantly more robust than MST. However, the difference between Alpino and Malt is not significant.

To summarize, our empirical evaluation shows that the grammar-driven system Alpino is rather robust across domains. It is the best performing system and it is significantly more robust than MST. In contrast, the transition-based parser Malt

scores the lowest across all domains, but its variation turned out not to be different from Alpino. Over all domains, MST is the most domain-sensitive parser.

Excursion: Lexical information Both kinds of parsing systems rely on lexical information when learning their parsing (or parse disambiguation) model. However, how much influence does lexical information have? To start examining this issue, we retrain all parsing systems by excluding lexical information. As all systems rely on a feature-based representation, we remove all feature templates that include words or stems and thus train models on a reduced feature space (original versus reduced space: Alpino 24k/7k features; MST 14M/1.9M features; Malt 17/13 templates).

The unlexicalized models are evaluated on the Wikipedia domains. The baseline is again 5-fold cross validation on the source domain (cdb). Obviously, absolute performance drops for all parsers. In more details, lexicalized versus unlexicalized baseline performance in LAS is, for each parser: Alpino $90.75 \rightarrow 89.36$, MST $83.63 \rightarrow 73.14$, Malt $79.95 \rightarrow 73.67$. Thus, as expected, performance drops to a higher degree for the data-driven parsers, but more for MST (-10.49) than for Malt (-6.28). In contrast, the variation in performance across domains (average domain variation) remains similar for most parsers, and is generally slightly smaller (with the exception of Malt): Alpino $adv = 0.81 \rightarrow 0.77$, MST $adv = -2.2 \rightarrow -0.44$, and Malt $adv = 0.59 \rightarrow 1.3$.

From the previous sections we know that the MST parser is the most domain-sensitive parser. The experiment presented in this section seems to suggest that this domain-sensitivity comes indeed from its high reliance on lexical information. When the lexical information is omitted, the MST parser suffers the most: Its absolute performance level drops to 73.14 LAS, even below the unlexicalized baseline of Malt. Moreover, MST scores below Malt on all Wikipedia domains when evaluated in this unlexicalized setting. Thus, even though Malt is the lowest scoring system in the lexicalized case, it seems that Malt is relying less on lexical information, and is thus less affected. In contrast, the grammar-driven parser Alpino suffers less from the missing lexical information.[7]

7 Conclusions and Future Work

We examined a grammar-based system coupled with a statistical disambiguation component (Alpino) and two data-driven statistical parsing systems (MST and Malt) for dependency parsing of Dutch. By looking at the performance variation across a variety of domains, we addressed the question of how sensitive the parsing systems are to the text domain. This, to gauge which kind of system is more affected by domain shifts, and thus more in need for adaptation techniques. We also proposed a new simple measure to quantify domain sensitivity.

The results show that the grammar-based system Alpino is the best performing system, and it is robust across domains. In contrast, MST, the graph-based

[7] Note that Alpino has still access to its lexicon here; for now we removed lexicalized features from the trainable part of Alpino, the statistical disambiguation component.

approach to data-driven parsing is the most domain-sensitive parser. The results for the transition-based parser Malt indicate that its sensitivity is limited, but it is generally (in absolute terms) among the lowest scoring systems. In general, data-driven systems heavily rely on the training data to estimate their models. This becomes apparent when we exclude lexical information from the training process, which results in a substantial performance drop for the data-driven systems, MST and Malt. The grammar-driven model was more robust against the missing lexical information. Grammar-driven systems try to encode domain independent linguistic knowledge, but usually suffer from coverage problems. The Alpino parser successfully implements a set of unknown word heuristics and a partial parsing strategy (in case no full parse can be found) to overcome this problem. This makes the system rather robust across domains, and, as shown in this study, significantly more robust than MST. This is not to say that domain dependence does not consitute a problem for grammar-driven parsers at all. As also noted by Zhang and Wang (2009b), the disambiguation component and lexical coverage of grammar-based systems are still domain-dependent. Thus, domain dependence is a problem for both types of parsing systems, though, as shown in this study, to a lesser extent for the grammar-based system Alpino. Of course, these results are specific for Dutch; however, it's a first step. As the proposed methods are independent of language and parsing system, they can be applied to another system or language.

Another research question examined in this study is how parsing performance correlates to simple statistical measures of the text. By looking at four measures, we could confirm the general result found by Zhang and Wang (2009a): different parsing systems have different sensitivity against statistical measures of the text. While they evaluated parsing systems for English, we here looked at dependency parsing for a freer word order language as Dutch. Both data-driven parsers show a high correlation to unknown word rate, while this is not the case for the grammar-based system. The highest correlation with parsing accuracy was found for the measure we added, sentence perplexity. This is true for both kinds of parsing systems, grammar-based and data-driven, but especially for the statistical parsers MST and Malt. This might first seem counterintuitive, as a grammar-based system usually suffers more from coverage problems. However, as already mentioned, Alpino successfully implements a set of unknown word heuristics to achieve robustness. For instance, on the 'bike winners list' sports domain, which we could identify through these simple statistical measures, Alpino and MST indeed exhibit a very different performance level, showing that the grammar-based system suffered less from the peculiarities of that domain.

In future, we would like to extend this line of work. It might be worth exploring more statistical measures of the text, to build a "domain detection" system for parsing. As for domain sensitivity, we would like to perform an error analysis, to examine why for some domains the parsers outperform their baseline and what are typical in-domain and out-domain errors.

References

Blitzer, John, Mark Dredze, and Fernando Pereira (2007), Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification, *In ACL*, Prague, Czech Republic.

Blitzer, John, Ryan McDonald, and Fernando Pereira (2006), Domain adaptation with structural correspondence learning, *In EMNLP*, Sydney, Australia.

Buchholz, Sabine and Erwin Marsi (2006), Conll-x shared task on multilingual dependency parsing, *In CoNLL*, pp. 149–164.

Daumé III, Hal (2007), Frustratingly easy domain adaptation, *In ACL*, Prague.

Dredze, Mark, John Blitzer, Pratha Pratim Talukdar, Kuzman Ganchev, Joao Graca, and Fernando Pereira (2007), Frustratingly hard domain adaptation for parsing, *Proceedings of the CoNLL Shared Task Session*, Prague, Czech Republic.

Gildea, Daniel (2001), Corpus variation and parser performance, *In EMNLP.*

Hara, Tadayoshi, Miyao Yusuke, and Jun'ichi Tsujii (2005), Adapting a probabilistic disambiguation model of an hpsg parser to a new domain, *Proceedings of the International Joint Conference on Natural Language Processing*.

McClosky, David, Eugene Charniak, and Mark Johnson (2006), Effective self-training for parsing, *In HLT-NAACL*, New York City, pp. 152–159.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič (2005), Non-projective dependency parsing using spanning tree algorithms, *In HLT and EMNLP*, pp. 523–530.

Nivre, Joakim and Ryan McDonald (2008), Integrating graph-based and transition-based dependency parsers, *In HLT-ACL*, Columbus, Ohio, pp. 950–958.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi (2007), Maltparser: A language-independent system for data-driven dependency parsing., *Natural Language Engineering* **13**, pp. 95–135, Cambridge University Press.

Oostdijk, Nelleke (2000), The Spoken Dutch Corpus: Overview and first evaluation, *In LREC*, pp. 887–894.

Plank, Barbara (2010), Improved statistical measures to assess natural language parser performance across domains, *Proceedings of LREC 2010*, Malta.

Plank, Barbara and Gertjan van Noord (2008), Exploring an auxiliary distribution based approach to domain adaptation of a syntactic disambiguation model, *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation (PE)*, Manchester.

Plank, Barbara and Gertjan van Noord (2010), Grammar-driven versus data-driven: Which parsing system is more affected by domain shifts?, *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground*, Uppsala, Sweden.

Ravi, Sujith, Kevin Knight, and Radu Soricut (2008), Automatic prediction of parser accuracy, *In EMNLP*, Morristown, NJ, USA, pp. 887–896.

Sagae, Kenji, Yusuke Miyao, and Jun'ichi Tsujii (2007), Hpsg parsing with shallow dependency constraints, *In ACL*, Prague, Czech Republic, pp. 624–631.

van Noord, Gertjan (2006), **A**t **L**ast **P**arsing **I**s **N**ow **O**perational, *In TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles*, Leuven, pp. 20–42.

Zhang, Yi and Rui Wang (2009a), Correlating natural language parser performance with statistical measures of the text, *In KI 2009*, Germany.

Zhang, Yi and Rui Wang (2009b), Cross-domain dependency parsing using a deep linguistic grammar, *In ACL-IJCNLP*, Suntec, Singapore, pp. 378–386.

To ish or not to ish?

Daphne Theijssen, Hans van Halteren, Tip Boonpiyapat, Anna Lohfink, Bas Ruiter, and Hans Westerbeek

Department of Linguistics, Radboud University Nijmegen

Abstract

In English, new adjectives can be coined by adding the suffix -ish. For instance, one can describe someone who acts like Arnold Schwarzenegger as *Schwarzeneggerish*. This paper investigates how the use of -ish is influenced by text characteristics (genre, formality) and author characteristics (gender, age). We used two corpora, the British National Corpus and the Blog Authorship Corpus. From our analyses of variance (ANOVAs) and logistic regression models, we learned that for the use of -ish it is probably more important what type of text you are writing than who you are. We also concluded that this type of research is seriously hampered by the absence of the kind of metadata needed for our type of research.

1 Introduction

Languages tend to be reasonably complete in the sense that they provide words for the things that people are likely to say to each other on a day-to-day basis. However, for those cases where one would need to say something not yet catered for, languages also provide means to form new words. Say one would like to express a quality for which there is no word, but the quality is well-represented by describing it as being like another quality or entity, e.g. an uncompromising, pushy and even violent attitude could be described as being *like Arnold Schwarzenegger*. This likening to existing words is fairly common, in fact so common that the English language provides the adjective-forming suffix -ish for it, so that the above description can be replaced by the single word *Schwarzeneggerish*.

In this paper, we investigate the use of this suffix as witnessed in corpus material.[1] We do not limit ourselves to really productive uses, but also include those words which have been created long ago and have managed to conquer a place in the dictionary. To be exact, we are interested in the representation of qualities by likening to something else (by use of -ish) and it does not matter if the actual formation of the -ish word was done by the current author or by, say, Shakespeare. We do limit ourselves to adjectives which have been formed by suffixation with -ish, excluding for instance words like *finish*, which does not contain the suffix -ish, but is a verb ending in *ish*. We also limit ourselves to those instances in which -ish bears the sense 'representing a likeness' (so not, e.g., *Finnish*).

[1] The research was carried out in the course 'Corpus-based Methods' in the research master Language and Communication, a collaboration between Radboud University Nijmegen and Tilburg University.

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

Our investigation focuses on the extralinguistic circumstances under which -ish words are being used. There are at least two aspects here that we want to consider. The first is the type of text in which the words are found. The Longman grammar of spoken and written English (Biber et al. 1999), e.g., tells us that we come across the phenomenon mostly in conversation and fiction, i.e. texts showing a more informal style. For corpora with metadata that have been designed with linguistic research in mind, such as the BNC (BNC Consortium 2007), we can use the genre classification to see whether this is indeed true.

The second aspect is the type of author that wrote the text. It has been remarked that the use of -ish is much more extensive among younger people. For instance, there is a Facebook fan page called "Adding *ish* onto the end of a word when describing something" with over 800,000 fans. And one can also imagine that a tendency for description by likening might be connected to the psychological makeup of the author and that there might be differences in use by men and women. Again, this can be investigated on the basis of corpus data, provided that the relevant metadata are available.

In summary, the research question addressed in this paper is the following: can the amount of use in a text of words formed with the suffix -ish, in the sense of a likeness, be shown to be dependent on a) genre or formality of the text and b) gender and/or age of the author of the text?

In the remainder of the paper, we first present related work (Section 2). Then we describe in detail which -ish words we have chosen to consider in our investigation (Section 3). In Section 4, we describe the collection of the data set on which we perform two statistical analyses. Section 5 shows our findings when applying analysis of variance (ANOVA), while in Section 6, we use logistic regression. A discussion is provided in Section 7, our main conclusion in Section 8.

2 Related work

A large part of the research on the suffix -ish has focussed on the methodology for measuring the productivity of suffixes in general (e.g. Nishimoto 2004, Plag 2006). The focus has mostly been on comparing different methods for measuring the productivity, and comparing the productivity of different suffixes. Plag (2006) showed that in the BNC, the range of different words to which -ish is added is not very large, compared to other suffixes like -ness and -ion. But relatively many were hapaxes, indicating that -ish is very suitable for the coinage of new words. Baayen (1994) showed that there is a clear relation between the productivity of various affixes and text type (genre). For the suffix -ly, for instance, he concluded that the text type and the author's individual preferences can overrule restrictions on the possible stems. The suffix -ish was not included in this paper.

Some researchers have focussed on the possible stems to occur with -ish. Byrd et al. (1986) argued that nouns with the suffix -ish tend to be short. They found no four-syllable word to occur with -ish. Spencer (2005) stated that -ish can be attached to whole phrases, e.g. *a why-does-it-have-to-be-me-ish expression*. Such instances are most likely to occur with very frequent and easily recognisable

phrases, he argued, but he was unaware of studies focussing on this type of suffixing. Prcic (1999) gave a nice overview of the possible stems preceding -ish, e.g. adjectives, nouns and numerals. He also discussed the dropping of a stem-final *e* (e.g. *blue - bluish*) and the doubling of a stem-final single consonant (*snob - snobbish*).
The combination of different suffixes, including -ish, has also been researched. The focus has either been on the order in which they can or cannot be attached (e.g. Hay and Plag 2004), their separability from the stem (e.g. Hay 2002), or their repetitive use (e.g. *boyishishness* in Plag and Baayen 2009). There is also some research on the history of the suffix -ish, for example in the works by Shakespeare (Neuhaus and Spevack 1975). The use of -ish to modify colours has also been the topic of study by a number of researchers (e.g. Moroney 2003, Rao and Lohse 1996). And there is also quite some work on children's acquisition of suffixes (e.g. Klibanoff and Waxman 2000).

Despite the fact that a lot of research has been carried out on the use of the suffix -ish, we are not aware of studies that examine the influence of characteristics of the speaker or writer (age, gender) and text (genre, formality) on the tendency to use -ish. The goal of the current paper is thus to provide such a study.

3 An inventory of *-ish* words

To investigate whether the use of -ish in the sense of likeness depends on extralinguistic characteristics, we needed an inventory of words containing the suffix in this sense. This section shows how we constructed this inventory and how we divided the words into different classes (treated separately in the statistical analyses).

3.1 Corpora and type extraction

Since the targeted suffix is productive, it was not possible to base our inventory on existing lexicons. Instead, we had to extract all potential instances from the corpora we planned to use. At the start of our investigation, we selected five corpora, listed in Table 9.1. Even though it later turned out that only BNC (BNC Consortium 2007) and BAC (Schler et al. 2006) were usable for analysis, we based the inventory on all five in order to keep the list as general as possible.

Corpus	Number of words
British National Corpus (BNC)	100,000,000
Wikipedia XML Corpus 2006 (WIKI)	350,000,000
Blog Authorship Corpus (BAC)	140,000,000
Caroline Tagg's Txt Msg Corpus (CorTxT)	189,000
Web-as-Corpus kool ynitiative UK (ukWaC)	2,000,000,000
Total	2,590,000,000

Table 9.1: Size of corpora at the basis of the inventory

We extracted all words from these corpora by splitting on white space, and kept those which ended in *ish* (ignoring word-final non-alphanumeric characters). Out of the 2.6 billion words present in the corpora, we extracted 4,025,285 tokens showing a total of 20,199 types.

3.2 Filtration

However, not all these 20,199 types were instances we want to consider. We automatically removed all URLs (477 types, especially frequent in the ukWaC corpus) with regular expressions specifying that the to-be-removed type contained strings like `www`, `http` and `.co.uk`. We also excluded the words (44 types) that consisted of four or fewer letters (e.g. *ish*, *yish*), making an exception for instance where the first character is a number (e.g. *6ish*). Finally, we used CELEX (Baayen et al. 1993) to remove the 210 types which were verbs (e.g. *finish*), common nouns (e.g. *parish*) or adjectives with an initial capital letter (e.g. *English*). The number of types we removed in this step may seem relatively small (731 of 20,199), but they account for over 90% (3,633,639) of the 4,025,285 tokens in the five corpora.

3.3 Manual classification

After this step of automatic removal, there were still 19,468 types left. We performed a pilot study to check the feasibility of manual identification of those types that can be split into a base and -ish, and where -ish does represent likeness. Three annotators, who were all non-native but fluent speakers of English, judged the relevance of the 1000 most frequent -ish words in the list of 19,468. The Kappa scores reached between the different pairs of annotators were low: -0.16, -0.04 and 0.44. We therefore had to conclude that the task proves too difficult.

The pilot study also made clear why this task was so difficult. There were many clear cases with recognisable stems, e.g. *foolish*, *greenish* and *silly-little-me-late-again-ish*. But to recognise other instances, one needed a vocabulary of a size that would have been daunting to many a native speaker, e.g. *priggish* and *marish*. This was also true for instances like *skittish*, which should not be included since -ish here had another sense than likeness. The need for vocabulary knowledge turned into a need for world knowledge when the type in question was capitalised: one needed to know that there is a singer called Björk to be able to approve *Bjorkish*, or that *Hammish* is just a less frequent spelling of the name *Hamish*.

3.4 Automatic classification

Having determined that manual annotation would lead to inconsistent, not well-analysable data, we decided to fall back on more trustworthy resources. We only kept those -ish words that on the basis of these resources could be classified as numeral-based (e.g. *6pmish*), name-based (e.g. *Beatles-ish*), noun-based (e.g. *bunnyish*) or adjective/adverb-based (e.g. *happy-ish*). A disadvantage of this procedure is that almost 80% (16,056) of the types could not be classified into the four classes. We thus fail to identify some of the more creative type such as

hahahahahahahahahahahahahahahahahahaha-ish. There is a risk that these creative types are exactly those characteristic for the language of younger people. However, this disadvantage is outweighed by the advantage of replicability.[2]

Numeral-based -ish words

In order to recognise numerals with the suffix -ish, we constructed two lexicons based on the findings in the corpora. The first was the *number lexicon* which consisted of cardinal numbers written out, e.g. *twelve* and *million*. Moreover, irregular ordinal numbers (e.g. *first* and *second*) and words related to times (e.g. *o'clock*, *quarter* and *p.m.*) were included. The second lexicon was the *unit lexicon*, consisting of units that we encountered in our data, e.g. *year-old*, *miles* and *hrs*. An -ish word was added to the numeral category if: (1) the first alphanumeric character of the word was a digit, and the stem consisted only of digits, non-alphanumeric characters and/or units in the unit lexicon (e.g. *13miles-ish*), or (2) the stem contained no capitals and consisted only of digits, non-alphanumeric characters, the coordinator *and* and/or numbers in the number lexicon (e.g. *two-thousandish*). Of the 20,199 types, 958 were found to have a numeral as base.

Name-based -ish words

For names, a different strategy was needed, as no comprehensive list of names could be built. We decided to include only the most certain types, being those capitalised forms with explicit marking of -ish, e.g. *Eeyore-ish* or *Jackass(ish)*. Types that occurred in the corpora in such a marked form could also be accepted when occurring without explicit marking. For such instances, the ratio between the number of occurrences with and without explicit marking should be below 50 in the five corpora. This for instance excluded *Kurdish*, which occurred 13,438 times, but only 1 time as *Kurd-ish*, and included *Schwarzeneggerish*, since this form and *Schwarzenegger-ish* both occurred once. Moreover, when changed to lower case, the stem should not be an adjective (e.g. excluding *Newish*), a common noun (e.g. excluding *Bullish*) or a numeral (e.g. excluding *Threeish*). This procedure led to a list of 1,001 types that are name-based.

Noun-based and adjective/adverb-based -ish words

For -ish applied to nouns, adjectives and adverbs, we only included those instances for which the base was present as such in CELEX (Baayen et al. 1993). For the base, we did allow inflected forms such as plural nouns and comparative/superlative adjectives and adverbs. Following Prcic (1999), we derived three potential bases from each -ish type: the actual form (e.g. *green* from *greenish*), the actual form with an *e* (e.g. *blue* from *bluish*) and the actual form but undoubling any double final consonant (e.g. *red* from *reddish*). We applied this procedure to all types without capitals and with at least one vowel.

[2]Our -ish inventory is available at `http://lands.let.ru.nl/~daphne/downloads`.

There were some bases which were ambiguous as to class, e.g. *chocolate*, which could be interpreted as a noun and an adjective/adverb according to CELEX. We disambiguated these manually: We preferred noun for stems that were intuitively hard to interpret as an adjective (e.g. *fool*) and that were materials (e.g. *copper*). The rest, including colours (e.g. *blue*, *blonde*) and directions (e.g. *west*, *up*), we preferred to label as adjectives/adverbs.
After dividing all these -ish words into the categories *noun-based* and *adjective/adverb-based*, we manually checked all noun-based types which had at least 5 instances in the corpora. We removed those words where -ish showed another sense than likeness, e.g. *stylish*, forms that were probably spelling errors, e.g. *finsish*, and uncapitalised names, e.g. *hamish*. The final set of noun-based -ish words contained 1,476 types, and that of adjectives/adverbs-based -ish words consisted of 708 types.

4 Collection of a data set

Having established four lists of words with the suffix -ish with the desired meaning, the next step was to prepare the corpus data for the analysis of the degree of their presence. For this we did not only need to count the selected words in each text, but also had to determine for the text the extralinguistic features we wanted to relate the degree of use of -ish words to: the genre/formality of the text, and the gender and age of the author. This section mainly describes how we determined these features, but returns at the end to the degree of use of -ish words.

4.1 Availability of metadata

As stated above, we employed five corpora to establish our inventory of -ish words (WIKI, CorTxT, BNC, BAC and ukWaC). However, only BNC and BAC provide the desired author information at the text level. The other three corpora do not contain metadata at all, Wikipedia and ukWaC being automatically crawled and lacking such data completely and CorTxT having been anonymised. The genre is provided in the BNC, but not for BAC. For the latter, we had to determine genre/formality in some way on the basis of the text itself.

For the statistical analyses, we thus limited ourselves to two corpora: BNC and BAC. They differ on a number of levels: Where the BNC contains only British English, the type of English in BAC is unknown. The BNC contains language from various sources, while the BAC contains only blogs from `blogger.com`. Most of the data in the BNC stems from the period 1985-1995, while most of the data in the BAC comes from 2000-2004. Because of these differences, we treated the use of -ish in the corpora separately in this paper.

4.2 Gender and age

For the BAC, all 19,320 texts are marked with the gender and age features. This is not the case for the BNC: although the documentation states that gender and age are present, we found that this is true for only 603 of the 4,050 texts. A case in

point is the spoken part of the corpus (908 texts), for which no speaker information is present. For instance, for the demographic part, it turned out that it is not the gender and age of the speakers that is available, but rather the gender and age of the respondent, the person who is walking around with the recorder. For the written part, 234 texts could not be used because they are marked as having mixed authorship and 2,304 because the author's age and/or gender is not documented. The author's age is classified into six classes in the BNC: 0-14, 15-24, 25-34, 35-44, 45-59 and 60+. The first of these, 0-14, contains very few texts, so we decided to merge it with the class 15-24, leaving us with five classes spanning all possible ages. The author ages in the BAC fall into three classes: 13-17, 23-27 and 33-48. Authors with ages outside these three ranges were not included in the corpus, in order to have clearly separable groups for the corpus' main goal: authorship recognition and profiling. Since we analysed the two corpora separately, we decided to use the two classification schemes as is and not to attempt to merge them.

4.3 Genre and formality

Genre

The genre feature is only present for the BNC. The full genre scheme in BNC's metadata distinguishes no fewer than 70 different genres. In order to have sufficiently large classes for analysis we merged the genres of the 603 remaining texts into four major classes: academic (101 texts), non-academic (115 texts), fiction (244 texts) and the rest (143 texts). The largest subgenres in the rest category were *misc* (57 texts), *biography* (49 texts), *religion* (13 texts) and *commerce* (12 texts).

Formality

As already mentioned, we expect genre to be mostly effective because the genres are linked to a level of formality. Since only the BNC contains information about genre, we needed a measure for the formality of the text, to replace genre when necessary. We used the Flesch reading ease score (Flesch 1948) as an approximation of the formality. This measure, together with other estimations of sentence complexity, is very commonly used in automatic genre classification (see for instance the overview in Luštrek 2007). The Flesch reading ease score is a number that usually varies between 0 and 100 (but it can be even higher or lower), and for which a higher score is found to correlate with simpler text. It is designed to go down when a text contains more words per sentence (longer sentences), and more syllables per word (longer words). To be exact, it is calculated as follows:

$$reading\ ease = 206.835 - 84.6 * (y/w) - 1.015 * (w/s)\ ,$$

where y is the number of syllables in the text, w is the number of words in the text and s is the number of sentences in the text.

We used GNU Style and Diction[3] to calculate the reading ease score for each

[3]See `http://www.gnu.org/software/diction/diction.html`.

text. We then used this score to establish the formality, being a binary feature which had the value *Informal* when the reading ease score was over a threshold and *Formal* otherwise. Based on the measurements for BNC (fiction having mean 85.1 and standard deviation 6.3 and academic plus nonacademic having mean 53.5 and standard deviation 9.8), we set the threshold at 73.0.
For the BNC, 97.5% of the academic and non-academic texts were considered *formal* in this measure, and 96.8% of the works of fiction *informal*. In the rest category, 68.7% of the texts were labelled *formal* and 31.3% *informal*.

4.4 Final selection

For the BAC we could use all 19,320 texts. For the BNC, as already mentioned, we removed all texts with insufficient metadata, all spoken texts and all texts with mixed authorship. Furthermore, we limited ourselves to recent English, removing texts from before 1985 (54 texts). The final set comprised 549 out of the 4,050 texts. Table 9.2 shows the average use of -ish words in the two sets.

	BNC			BAC		
Type of base	N	mean	stdev	N	mean	stdev
adjectives/adverbs	549	25.6	75.0	19,320	16.2	122.6
names	549	0.3	2.7	19,320	2.4	43.1
nouns	549	69.5	134.0	19,320	49.3	214.1
numerals	549	1.7	20.6	19,320	15.7	125.7

Table 9.2: Number of texts (N), mean number of -ish words per million (mean) and standard deviation (stdev) for the different categories and corpora.

The standard deviations show that there is a lot of variation between the texts. Moreover, we see that noun-based -ish words are the most common. Considering the four categories as one category (disregarding the categorisation) could yield results that are mostly based on the noun-based -ish words. To avoid this, we kept the four categories apart in the statistical analyses. Because the use of numeral-based and name-based -ish words is very small in the BNC, we excluded the analysis of these two categories for the BNC. For the BAC, we kept all four categories.

5 Analysis of variance (ANOVA)

We used analysis of variance (ANOVA) to test whether the average value of the dependent variable (the use of -ish per million words) significantly differs between certain groups, taking into account individual variation within the groups.

5.1 Method

We employed a factorial analysis of variance (ANOVA) with three fixed factors (age, gender and genre/formality).[4] Only the interaction between gender and age was included. Since the dependent variable needs to have a distribution that is (close to) normal, we excluded all instances without an -ish word (i.e. having value *0*) and for the remaining instances took the log of the number of instances per million.
In ANOVA, the log of the use of -ish per million is calculated as follows:

$$X_{ijkl} = \mu + \alpha_i + \beta_j + \alpha\beta_{ij} + \gamma_k + U_{ijkl} \ ,$$

in which X is the use of -ish per million for individual text l with a writer aged i and having gender j, having genre/formality k. The symbol μ represents the overall average, α_i the average deviance from μ in this age group, β_j the average deviance from μ in this gender group, $\alpha\beta_{ij}$ the average deviance from μ in this age and gender group, γ_k the average deviance from μ in this genre/formality group, and U_{ijkl} the individual deviance from μ. In order to establish the effect of the factors, ANOVA calculates the total sum of squares of the deviances from the average μ.

5.2 Results

The ANOVAs for noun-based and adjective/adverb-based -ish words in the BNC can be found in Tables 9.3 and 9.4. They show that there are no significant effects for the -ish words with an adjective or adverb as base. For the noun-based -ish words, gender, genre and formality have a significant effect on the use of -ish per million words. When we look at the average values, we see that females use noun-based -ish words much more frequently than men: on average, females use 131 noun-based -ish words per million (in the texts that contained at least one occurrence), males only 96 per million. With respect to genre, the presence of -ish is smallest in academic and nonacademic texts (61 per million), and largest in fiction (141 per million). The same effect is found for formality: formal texts have relatively few -ish words per million (80 per million), and informal texts relatively many (128 per million). For both adjectives and nouns, however, there is still a lot of variance unexplained, seeing the large sums of squares (SS) for the residuals.
The ANOVAs for the use of numeral-based, name-based, noun-based and adjective/adverb-based -ish words in the BAC set can be found in Tables 9.5 and 9.6. The formality of the text only has a significant effect on the use of numeral-based and adjective/adverb-based -ish words. It has the same pattern as the noun-based -ish words in the BNC: on average, formal texts show fewer occurrences of -ish words (237 per million for numerals, 195 per million for adjectives/adverbs) than informal texts (283 per million for numerals, 230 per million for adjectives/adverbs). Moreover, we find a (near) significant effect of age range

[4] We used the function `aov()` in the `stats` package in `R`.

	Noun				Adj/Adv		
	df	SS	F		df	SS	F
Gender	1	12.3	20.0	***	1	0.4	0.7
AgeRange	4	4.2	1.7		4	1.0	0.4
Genre	3	18.3	9.9	***	3	1.3	0.7
Gender:AgeRange	4	2.7	1.1		3	1.5	0.8
Residuals	342	210.9			166	105.9	

Table 9.3: ANOVA for noun-based and adjective/adverb-based use of -ish in BNC texts, using *genre*, *** p<0.001 ** p<0.01 * p<0.05 · p<0.10.

	Noun				Adj/Adv		
	df	SS	F		df	SS	F
Gender	1	12.3	19.1	***	1	0.4	0.7
AgeRange	4	4.2	1.6		4	1.0	0.4
Formality	1	6.0	9.3	**	1	0.1	0.1
Gender:AgeRange	4	4.3	1.7		3	1.7	0.9
Residuals	344	221.5			168	107.0	

Table 9.4: ANOVA for noun-based and adjective/adverb-based use of -ish in BNC texts, using *formality*, *** p<0.001 ** p<0.01 * p<0.05 · p<0.10.

for all types in this corpus, whereas there were no significant effects of age range found in the BNC data. The youngest bloggers, aged 13 to 17, use -ish words most frequently for all base categories except names. For name-based -ish words, the oldest bloggers, aged 33 to 48, are the most frequent users, although the difference is only near significance. As with the BNC, the high residuals show that a large part of the variance remains unexplained.

	Numeral				Name			
	df	SS	F		df	SS	F	
Gender	1	0.4	0.2		1	0.2	0.1	
AgeRange	2	138.8	41.5	***	2	11.2	3.0	·
Formality	1	13.9	8.3	**	1	1.6	0.9	
Gender:AgeRange	2	4.1	1.2		2	3.1	0.8	
Residuals	1079	1804.9			285	530.8		

Table 9.5: ANOVA for numeral-based and name-based use of -ish in BAC texts, *** p<0.001 ** p<0.01 * p<0.05 · p<0.10.

	Noun				Adj/Adv			
	df	SS	F		df	SS	F	
Gender	1	0.0	0.0		1	0.1	0.0	
AgeRange	2	85.6	36.6	***	2	78.8	25.7	***
Formality	1	0.8	0.7		1	7.6	5.0	*
Gender:AgeRange	2	4.6	2.0		2	2.5	0.8	
Residuals	3333	3901.7			1384	2125.8		

Table 9.6: ANOVA for noun-based and adjective/adverb-based use of -ish in BAC texts, *** p<0.001 ** p<0.01 * p<0.05 · p<0.10.

6 Logistic regression

We also addressed our research question by considering it as a classification problem: we used a modelling or machine learning technique to predict whether the number of -ish words in the text is above-average or not. So, using the per-million-counts for each text in our BNC data set, we established the average number of noun-based and adjective/adverb-based -ish words per million, and for each text checked whether they were above average or not. The same was done for the BAC data, also for the name-based and numeral-based -ish words.

Because the texts in the BAC were relatively short (7,004 words on average), over 76% (14,789) of the texts in the BAC did not contain an -ish word at all. The texts in the BNC set were over five times longer on average (35,716 words), and consequently only 29% (159) of the texts lacked an occurrence of -ish. As a result, there was a higher imbalance between the above and below average use of -ish in the BAC set than in the BNC set.

We experimented with a number of techniques, including Naive Bayes, Ripper, C4.5 and Logistic regression. The first three, being classifiers, suffered from the imbalance in the BAC. The high majority baseline (86% to 98%) made it difficult for the algorithms to learn patterns that were able to improve this baseline. Logistic regression is much more robust with respect to class imbalance (Owen 2007). We thus limit ourselves to logistic regression in this paper.

6.1 Method

We modelled the use of -ish as depending on the values assigned to the same features as in the previous section: the age of the writer, his/her gender, the interaction between the two, and the genre or formality of the text.

In logistic regression modelling, a regression function is established that fits the data matrix best. It yields the log of the odds that the use of -ish in the text in question (I) is 1 (above average) rather than 0 (below average):

$$\texttt{logit}[p(I = 1)] = \alpha + \beta X$$

where α is the model intercept, X are the feature values and β are the coefficients. The coefficients β can be understood as the weights assigned to the features by the model, where positive values increase, and negative values decrease, the odds that there is above average use of -ish. The optimal values for α and β were estimated using Maximum Likelihood Estimation[5].

The log odds were used to establish the quality of the model. In this paper, we use the area under the ROC curve (AUC), which gives the probability that the regression function, when randomly selecting a positive (above average use of -ish) and a negative (below average use of -ish) instance, outputs a higher log odds for the positive instance than for the negative instance.[6]

6.2 Results

The AUC reached by the models built on the BNC data can be found in the top two rows of Table 9.7. Although their values are not really high, they are sufficient to allow an analysis of the coefficients in the model, presented in Table 9.8. The results are similar to what we found for the ANOVAs in Section 5. Writers are more likely to use a noun-based or an adjective/adverb-based -ish word when writing a work of fiction (the coefficients 0.93 and 0.57 are both positive), and less likely when writing an academic text (-1.33 and -1.41). Informal texts are more likely to show above average use of -ish (1.11 and 1.04) than formal texts.

	Numeral	Name	Noun	Adj/Adv
BNC model with genre			0.751	0.687
BNC model with formality			0.708	0.658
BAC model with formality	0.592	0.547	0.536	0.561

Table 9.7: AUC reached by the regression models

The AUCs for the four regression models built on the BAC data are presented in the bottom row of Table 9.7. The scores are very low; the models are only slightly better than chance (AUC=0.5). The regression coefficients are presented nonetheless in Table 9.9. As in the ANOVAs, there are many more significant effects than in the BNC. The formality of the text is significant for numeral-based and adjective/adverb-based -ish words: informal texts tend to have more above average use of -ish for these categories than formal texts. There are also some significant effects of gender and age. For age range 13-17, we see that the use of noun-based -ish words is likely to be lower than for the other age ranges. This is the opposite of what we saw in the previous section, where the average use of noun-based -ish words was highest for the youngest bloggers. This difference, the small values of the coefficients and the low AUC show that it may be unadvisable to draw strong conclusions from the coefficients in this model.

[5] We used the function `glm()` in the `stats` package in `R`.

[6] We used the function `somers2()` created in `R` by Frank Harrell.

Feature	Noun		Adj/Adv	
(Intercept)	-0.68 **	-1.08 ***	-0.70 **	-1.23 ***
Female	0.34	0.35	0.02	0.04
Age 0-24	-14.56	-14.91	-1.02	-1.29
Age 25-34	0.12	0.18	-0.74	-0.70
Age 45-59	-0.23	-0.36	-0.04	-0.14
Age 60+	-0.13	-0.22	0.34	0.29
Academic	-1.33 ***		-1.41 ***	
Fiction	0.93 ***		0.57 *	
Nonacademic	-0.63 *		-0.61 *	
Informal		1.11 ***		1.04 ***
Female, Age 0-24	15.38	15.30	-13.71	-13.72
Female, Age 25-34	0.14	-0.10	1.10	0.88
Female, Age 45-59	0.08	0.31	-0.08	0.03
Female, Age 60+	0.44	0.59	-0.35	-0.29

Table 9.8: Coefficients of BNC regression models,*** p<0.001 ** p<0.01 * p<0.05 · p<0.10.

Feature	Numeral	Name	Noun	Adj/Adv
(Intercept)	-3.73 ***	-4.01 ***	-1.72 ***	-3.11 ***
Female	0.40 ***	0.00	0.15 *	0.43 ***
Age 13-17	0.21 *	-0.36 *	-0.20 **	0.08
Age 33-48	-0.37 *	-0.42	-0.13	-0.02
Informal	0.68 ***	0.04	-0.03	0.24 **
Female, Age 13-17	-0.09	-0.02	-0.05	-0.07
Female, Age 33-48	-0.14	0.11	-0.20	-0.05

Table 9.9: Coefficients of BAC regression models,*** p<0.001 ** p<0.01 * p<0.05 · p<0.10.

7 Discussion

For the BNC, we have been able to confirm the finding put forward in the literature that formality is a major factor in the amount of use of -ish. Both statistical analyses showed that -ish is applied to noun bases significantly less in more formal texts than in more informal texts. This effect could be shown both on the basis of the genre classification provided in the metadata and on the basis of an automatically derived substitute, our formality measure based on the Flesch reading ease score. For adjective/adverb-based -ish words, we only found an effect of genre and formality in the logistic regression models, but it showed the same pattern. As to the gender and age of the author, only ANOVA discovered a significant influence on the application of -ish in the BNC, but only for those with noun bases. It seems that females more frequently use noun-based -ish than males, but this is not confirmed by the logistic regression model. It could be that our inventory of -ish words is too limited to discover more (clear) differences between age and gender groups, since

the more creative types have been left out of consideration.

For the BAC, only the application of -ish to numeral and adjective/adverb bases could be shown to be significantly influenced by formality with both methods. For the age and gender of the writer, it appears there are significant influences. There seems to be a significant effect of age range on the use of -ish for all four bases, although it is not confirmed by the logistic regression model for adjectives/adverbs. Moreover, some coefficients in the logistic regression model showed different effects than the average values used for the ANOVAs. It seems that these observations are not unequivocal, as both methods reported low model quality (by residuals and AUC). Apparently, the factors we included in the analyses are not enough to explain the variance in the data; there is still much individual variation left.

It may even be that the influences we found are only indirect, being an artifact of specific gender/age groups writing about specific topics, which we could not take into account due to the absence of metadata. We have thus observed (yet again) that for an investigation into linguistic variation, one not only needs large amounts of textual data, but comprehensive metadata as well. The currently available corpora tend to fail in at least one of these demands. The BNC, although in principle well-supplied with metadata, in the end provides relatively few texts for which the desired metadata is present, potentially too few to be able to show significance of the relevant influences. The BAC, although containing larger amounts of text, only provides very limited metadata, while larger corpora, such as ukWaC, lack the kind of metadata that are needed for linguistic research altogether. Here, factorial statistical analysis becomes impossible from the start.

8 Conclusion

In this paper, we investigated whether the amount of use in a text of the English suffix -ish, in the sense of likeness, is dependent on genre/formality of the text and the gender and age of the author. We did this by applying two types of statistical analysis, ANOVA and logistic regression, to two corpora, a selection of written texts from the British National Corpus (BNC) and the Blog Authorship Corpus (BAC). The large unexplained variance of the ANOVA (the residuals) and the low confidence of the logistic regression models (the AUC) showed that we could only partly explain the use of -ish. The results seem to indicate that, as to the use of -ish, it is probably more important what you are writing than who you are.

For our study of the suffix -ish, we have been partially able to circumvent the lack of metadata (at least for the BAC) by replacing the genre/formality feature by an automatically derivable substitute based on the Flesch reading ease score. But this was only one feature, representing only one dimension of the in principle much richer genre feature. Given the potential of the extraction of huge corpora from the internet, but their lack of metadata, we think that a good way forward would be to attempt to induce some classification similar to the metadata in BNC automatically from the text. Although classifying gender and age appear a bridge too far at the moment, features like genre and domain ought to be feasible. Of course, automatic induction of such metadata has many possible drawbacks and

pitfalls. Still, we believe that such automatic induction is a fruitful avenue of future research. Once we manage to generate these metadata automatically, there will be a basis for a much larger number of investigations into linguistic variation than is possible now.

References

Baayen, R. Harald (1994), Derivational productivity and text typology, *Journal of Quantitative Linguistics* **1** (1), pp. 16–34, Routledge.

Baayen, R. Harald, Richard Piepenbrock, and Hedderik van Rijn (1993), *The CELEX Lexical Database (CD-ROM)*, Linguistic Data Consortium, University of Pennsylvania, Philadelphia, USA.

Biber, Douglas, Stig Johansson, Geoffrey Leech, Susan Conrad, and Edward Finegan (1999), *Longman grammar of spoken and written English*, MIT Press.

BNC Consortium (2007), *The British National Corpus, version 3 (BNC XML Edition)*, Oxford University Computing Services. http://www.natcorp.ox.ac.uk/.

Byrd, Roy J., Judith L. Klavans, Mark Aronoff, and Frank Anshen (1986), Computer methods for morphological analysis, *Proceedings of the 24th annual meeting of the Association for Computational Linguistics*, pp. 120–127.

Flesch, Rudolph (1948), A new readability yardstick, *Journal of applied psychology* **32** (3), pp. 221–233.

Hay, Jennifer (2002), From speech perception to morphology: Affix ordering revisited, *Language* **78** (3), pp. 527–555, Linguistic Society of America.

Hay, Jennifer and Ingo Plag (2004), What constrains possible suffix combinations? On the interaction of grammatical and processing restrictions in derivational morphology, *Natural Language & Linguistic Theory* **22** (3), pp. 565–596, Springer.

Klibanoff, Raquel S. and Sandra R. Waxman (2000), Basic level object categories support the acquisition of novel adjectives: Evidence from preschool-aged children, *Child Development* **71** (3), pp. 649–659, Blackwell Publishers.

Luštrek, Mitja (2007), Overview of automatic genre identification, *Technical Report IJS-DP-9735*, Jožef Stefan Institute, Department of Intelligent Systems, Jamova 39, 1000 Ljubljana, Slovenia.

Moroney, Nathan (2003), Unconstrained web-based color naming experiment, *Proceedings of the SPIE*, pp. 36–46.

Neuhaus, H. Joachim and Marvin Spevack (1975), A Shakespeare dictionary (SHAD): Some preliminaries for a semantic description, *Computers and the Humanities* **9** (6), pp. 263–270, Springer.

Nishimoto, Eiji (2004), Defining new words in corpus data: Productivity of English suffixes in the British National Corpus, *26th Annual Meeting of the Cognitive Science Society (CogSci 2004)*.

Owen, Art B. (2007), Infinitely imbalanced logistic regression, *The Journal of Machine Learning Research* **8**, pp. 773, MIT Press.

Plag, Ingo (2006), Productivity, *Handbook of English Linguistics* pp. 537–556.

Plag, Ingo and R. Harald Baayen (2009), Suffix ordering and morphological processing, *Language* **85** (1), pp. 109–152, Linguistic Society of America.

Prcic, T. (1999), The treatment of affixes in the 'big four' EFL dictionaries, *International Journal of Lexicography* **12** (4), pp. 263, Oxford University Press.

Rao, A. Ravishankar and Gerald L. Lohse (1996), Towards a texture naming system: Identifying relevant dimensions of texture, *Vision Research* **36** (11), pp. 1649–1669, Elsevier.

Schler, Jonathan, Moshe Koppel, Shlomo Argamon, and James Pennebaker (2006), Effects of age and gender on blogging, *AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*.

Spencer, Andrew (2005), Word-formation and syntax, *Handbook of word-formation*, Springer, pp. 73–97.

Emotion Classification in a Serious Game for Training Communication Skills

Frederik Vaassen and Walter Daelemans

CLiPS Research Center, University of Antwerp

Abstract

We describe the natural language processing component of a new serious gaming project, *deLearyous*, which aims at developing an environment in which users can improve their communication skills by interacting with a virtual character in (Dutch) written natural language. The virtual characters' possible dialogue paths are defined by Leary's Rose, a framework for interpersonal communication. In order to apply this framework, input sentences must be classified into one of four possible "emotion" classes.

We tried to carry out this emotion classification task using several machine learning algorithms. More specifically, classification performance was measured using TiMBL –a memory-based learner–, a Naïve Bayes classifier, Support Vector Machines and Conditional Random Fields. Training was done on a relatively small dataset of manually tagged sentences. A large number of different features was extracted from the dataset, and a good feature subset was selected using a combination of a genetic algorithm and various filter metrics.

We achieved the best results using the memory-based learner TiMBL, using a combination of word unigrams, lemma trigrams and dependency structures. With this setup, 52.5% of the sentences were classified into the correct emotion quadrant, which is a significant improvement over the statistical baseline (25.15%) and over the scores achieved with a pure bag-of-words approach (41.6%).

1 Introduction

1.1 *deLearyous*

Interpersonal communication is rarely simple. Our conversation partners may hold different opinions from our own, they may not be in the mood to listen to what we have to say, or they may simply not like us very much. Yet, we would still like to get our point across, preferably without disrupting our relationship with whomever it is we are talking to.

Communication skills are especially important in a professional context. The outcome of negotiations, for instance, may have a big impact on the future of a company. Likewise, an employer can avoid many complications if he manages to tell his employees about potentially unpopular management changes in a diplomatic way. For this reason, many companies invest in communication training for their employees. This training is usually done by acting out short scenarios with specialized actors. Hiring these actors, however, is a significant expense, and the

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

time which employees can spend on training their communication skills is limited by the company's budget or the actor's schedule.

It is to counter these two downsides that the *deLearyous* project was devised. The goal of the project is to develop a serious video game that can be used to assist in the training of interpersonal communication skills. The video game will present the player with a virtual autonomous character with whom he can interact by typing in natural language sentences (in Dutch). The virtual character will then reply in a way that fits the user's input and the underlying communication framework. By conversing with the virtual character and figuring out what conversation tactics work best to achieve their goals, players will be able to improve their own communication skills without the constraints that working with real actors imposes.

The *deLearyous* project is a co-operation between the e-Media Research Lab at Groep T in Leuven, the CLiPS Research Center at the University of Antwerp and Opikanoba, a company specialized in e-learning. The e-Media Research Lab will develop the dialogue manager as well as the audio and video modules, while Opikanoba will write the scenario for the virtual character. We, at CLiPS, will focus entirely on the Natural Language Processing component of the project, which is also the focus of this paper.

1.2 Leary's Rose

Several frameworks have been developed to describe the dynamics involved in interpersonal communication. The framework used in the *deLearyous* project is the *Interpersonal Circumplex*, also known as *Leary's Rose* (Leary 1957).

Leary's Rose (Figure 1) has both a descriptive and a predictive function: it can be used to position any participant in a discussion according to his state of mind and behavior, but it can also be used to predict (to a certain extent) the conversation partners' reaction to the speaker's behavior.

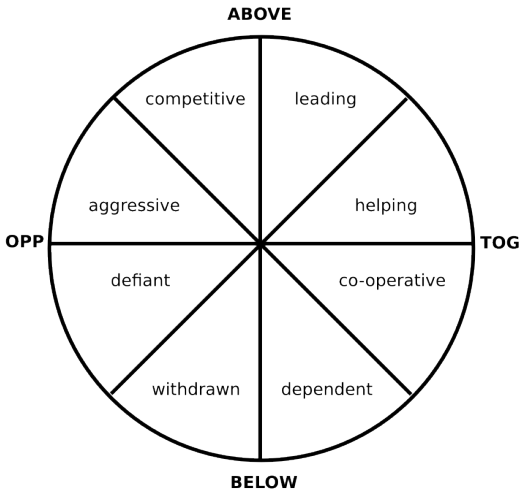


Figure 1: Leary's Rose

The Rose is defined by two axes: the *above-below* axis (vertical), which tells us whether the speaker is being dominant or submissive towards the listener; and the *together-opposed* axis (horizontal), which says something about the speaker's willingness to co-operate with the listener. The axes divide the Rose into four quadrants, and each quadrant can again be divided into two octants. Below are a few example sentences with their corresponding position on Leary's Rose.

- My name is John, how can I be of assistance? - **helping**
- How do you suggest we continue from here? - **dependent**
- That's not my fault, administration's not my responsibility! - **defiant**
- If you're going to be rude there's no use in continuing this conversation! - **aggressive**

Leary's Rose also allows us to predict what position the listener is most likely going to take in reaction to the way the speaker positions himself.

Two types of interactions are at play in Leary's Rose, one of complementarity and one of similarity:

- *above*-behavior triggers a response from the *below* zone and vice versa. When addressed by someone acting submissive, the listener's instinctive reaction will be to act more dominant and to take the lead, and vice versa.
- *together*-behavior triggers a response from the *together* zone, while *opposed*-behavior triggers a response from the *opposed* area of the Rose. When the speaker is being friendly towards the listener, the listener will be more likely to respond in kind, while unfriendly behavior is likely to elicit an unfriendly response.

The speaker can thus influence the listener's emotions (and consequently, his response) by consciously positioning himself in the quadrant that will likely trigger the desired reaction. Should the speaker wish to draw the listener out of the "aggressive" octant towards the "co-operative" octant, for instance, he would have to position himself in the *above-together* quadrant to gradually coax the listener to a more favorable disposition.

A first and crucial step in the development of *deLearyous* is making it possible to automatically detect the player's position on Leary's Rose. Since the player will interact with the virtual character by typing in natural language sentences, it is from these sentences that the "emotion" information is to be extracted. The emotion classification task will be the focus of the rest of this paper.

2 Related Work

To our knowledge, no work has yet been done specifically on the automatic classification of sentences based on Leary's Rose or on other incarnations of the Interpersonal Circumplex. There has however been quite a bit of research in the broader areas of automatic sentiment and emotion classification.

The techniques that have been used for sentiment and emotion classification can roughly be divided into pattern-based methods and machine-learning methods. An often-used technique in pattern-based approaches is to use pre-defined lists of keywords which help determine the instance's overall sentiment orientation or emotion contents. The AESOP system by Goyal et al. (2010), for instance, attempts to analyze the affective state of characters in fables by identifying affective verbs and by using a set of projection rules to calculate the verbs' influence on their patients. Balahur et al. (2010) evaluate several sentiment-annotated lexical resources (including MicroWNOp, WordNet Affect, SentiWordNet and their own JRC Tonality resource) on a set of newspaper quotes by computing sentiment in a window around the target of the quote.

Another possible approach is to let a machine learner determine the appropriate sentiment/emotion class. Mishne (2005) and Keshtkar and Inkpen (2009), for instance, attempt to classify LiveJournal posts according to their mood using Support Vector Machines trained with frequency features (word counts, POS-counts), length-related features (length of posts/sentences/...), semantic orientation features (using WordNet to calculate the distance of each word to a set of manually classified keywords) and special symbols (emoticons). Tsur et al. (2010) developed a system to recognize sarcasm in user opinions. They compiled feature vectors using punctuation-based features and patterns of high-frequency vs. content words, and applied a k-NN-like approach for classification.

Finally, Rentoumi et al. (2010) posited that combining the rule-based and machine learning approaches can have a positive effect on classification performance. By classifying strongly figurative examples using Hidden Markov Models while relying on a rule-based system to classify the mildly figurative ones, the overall performance of the classication system is improved.

3 Methodology

We have chosen to use a machine learning approach to try to automatically position Dutch sentences on Leary's Rose. Starting from a training set of sentences labeled with their position on the Rose, a machine learner should be able to pick up on cues that will allow the classification of new sentences into the correct emotion class.

Our approach falls within the domain of text categorization (Sebastiani 2002), which focuses on the automatic classification of text into predefined categories. Most text-categorization systems first extract features (terms, n-grams, ...) from a set of pre-classified documents. From these features, they then select those that are most helpful in predicting the document's category. This new feature subset is used to train a machine learner, which will then be able to classify new documents into the correct class. Text categorization has been used successfully for a wide array of applications, including document filtering, categorization of web content and authorship attribution (Luyckx and Daelemans 2005).

An important advantage of the machine learning approach compared to pattern-based approaches is that machine learners are able to take advantage of complex interactions between features, interactions which are often impossible to capture using handcrafted patterns. Since there are no easily identifiable keywords or syntactic structures that are consistently used with a position on Leary's Rose, using a machine learning approach is a logical choice for this emotion classification task.

3.1 Data

We compiled a small dataset of 339 Dutch sentences labeled according to their position on Leary's Rose. A large part of these sentences were taken from *Beïnvloed anderen, begin bij jezelf* (van Dijk 2000), a book specifically describing the workings of the Rose. Other sentences originated from scenarios which were specifically written for this purpose by Opikanoba and colleagues at CLiPS. The resulting dataset is relatively well balanced across the four quadrants of the framework:

	OPP_B	OPP_A	TOG_B	TOG_A
# of instances	80	82	96	81

3.2 Feature Extraction

From this dataset we extracted a wide range of different features. The sentences were first parsed with Tadpole, a Dutch language parser (van den Bosch et al. 2007), which allowed us to extract linguistic information such as word tokens, lemmas, part-of-speech tags, syntactic functions and dependency structures. The actual feature vectors were then generated on the basis of this linguistic information by using a "bag of n-grams" approach, i.e. by constructing n-grams (unigrams, bigrams and trigrams) of each feature type (e.g. n-grams of word tokens, n-grams of part-of-speech tags...) and by counting for each n-gram in the

training data how many times it occurs in the current instance. Additionally to these n-gram counts, we also included punctuation counts, average word length and average sentence length.

3.3 Feature Subset Selection, Parameter Optimization and Genetic Algorithms

We reduced the dimensionality of the resulting feature vectors by selecting subsets of informative features using a variety of filter metrics –specifically gainratio, infogain, the Gini coefficient and χ^2.

Since it is possible to vary the feature types included in the feature vectors as well as the filter metrics for subset selection and the number of features in the selected subset, testing out all possible combinations of these parameters would be a prohibitively time-consuming task[1] A solution to this problem is to use a genetic algorithm (we used Pyevolve[2], a genetic algorithm for Python) to try out different combinations of feature types, filter metrics and learner parameters for each individual learner, while maximizing the learner's classification accuracy[3].

3.4 Classification

The actual classification is done with one of four learners, TiMBL –a memory-based learner based on the k-NN algorithm– (Daelemans and van den Bosch 2005), a Naïve Bayes classifier as implemented in the Orange machine learning package for Python[4], a multiclass implementation of Support Vector Machines (SVM-multiclass[5]) and CRFsuite (Okazaki 2007), an implementation of Conditional Random Fields. Using a Naïve Bayes classifier and SVMs seemed like a natural choice for this type of classification problem, as these techniques have shown their worth in similar classification tasks in the past. TiMBL was co-developed by CLiPS and has turned out to be very useful for a wide range of NLP tasks, and it is also known to perform well on problems with small datasets. A CRF learner may seem like a less logical choice since the feature vectors we constructed no longer represent a sequence, but we have found CRFs to be surprisingly effective and have decided to include them in this overview.

[1] Exhaustively exploring subsets of up to three feature type combinations with TiMBL –without varying the learner parameters– would take approximately two weeks.

[2] *Pyevolve 0.6* by Christian S. Perone. - `http://pyevolve.sourceforge.net` (last visited on May 6th, 2010)

[3] The genetic algorithm was run for 20 generations using a crossover rate of 80% and a mutation rate of 1%. We used Pyevolve's default crossover method (one point crossover) and the "1D-list allele mutator". The parameters varied for TiMBL were the value of k, the weighting metric and the distance metric. For SVMs, we varied the tradeoff between margin and training error and the type of Kernel function. See sections 3.4 and 4.4 for a more detailed description of the classifiers and their parameters.

[4] *Orange 2.0b*, Laboratory of Artificial Intelligence, Faculty of Computer and Information Science, University of Ljubljana - `http://www.ailab.si/orange/` (last visited on May 6th, 2010)

[5] *SVM-multiclass 2.20* by Thorsten Joachims - `http://svmlight.joachims.org/svm_multiclass.html` (last visited on May 6th, 2010)

The instances were classified into one of the four quadrants of Leary's Rose and results were calculated on the basis of 10-fold cross validation.

4 Results

4.1 Baseline

The statistical baseline for this 4-class classification problem, taking into account the slight imbalances in the class distribution, is 25.15%. An additional and more useful baseline is the performance of each learner when using only token unigrams without any kind of feature subset selection. These baseline results are illustrated in Table 10.1.

	TiMBL	Naïve Bayes	SVM	CRF
statistical baseline		25.15%		
baseline accuracy	41.6%	41.6%	26.8%	44.2%

Table 10.1: Baseline scores using only token unigrams

4.2 Performance

Table 10.2 gives an overview of the top scores that we managed to achieve with each of the four learners, i.e. using the combination of features and learner parameters that was determined to give the best accuracy by the genetic algorithm. The "features" column indicates the types of features that have been used:

- w - word tokens
- l - lemmas
- c - characters
- d - dependency groups
- wlc - average word length based on the number of characters in a word
- wls - average word length based on the number of syllables in a word

The numbers appended to the feature types indicate the size of the n-grams used (1 - unigrams, 2 - bigrams, 3 - trigrams).

The parameters for each learner were determined using the genetic algorithm. TiMBL used a k-value of 1, gainratio for weighting and the dot-product metric as the distance metric. SVMs were used with standard parameters, except for the trade-off between training error and margin, which was set to 0.6[6].

All learners outperform the statistical baseline by a large margin, but the results of the memory-based learner TiMBL are especially interesting as they also signif-

[6]SVMs appear to be extremely sensitive to parameter changes, and there is a lot of interaction between learner parameters and the feature set used for training. We believe that the SVM scores can still be improved given a more stable dataset and a more thorough search through the parameter/feature space.

	features	# of feats	filter metric	accuracy	stdev.	F-score (macro avg.)
TiMBL	w1, l3, d	1000	Gini	52.5%	10.5%	52.2%
Naïve Bayes	l1, w3, wls	all	n/a	46.6%	8.8%	46.8%
SVM	c1, l3, wlc	250	infogain	35.7%	11.3%	35.7%
CRF	w1, wlc	500	infogain	47.8%	11.0%	47.7%

Table 10.2: Top results per learner

icantly improve on the second baseline[7], which uses token unigrams only. 52.5% of the sentences are assigned the correct quadrant in Leary's Rose.

Table 10.3 shows the detailed class scores for the TiMBL experiment, while Table 10.4 shows the confusion matrix.

	precision	recall	F-score
TOG_A	0.53	0.49	0.51
OPP_A	0.55	0.48	0.51
OPP_B	0.55	0.53	0.54
TOG_B	0.48	0.61	0.53

Table 10.3: Class scores for TiMBL

	true TOG_A	true OPP_A	true OPP_B	true TOG_B
TOG_A	43	17	8	19
OPP_A	17	45	18	13
OPP_B	9	12	44	18
TOG_B	12	8	10	46

Table 10.4: Confusion matrix for TiMBL

Performance is relatively uniform over all classes and there seem to be no significant trends in the classification errors. This also makes the analysis and correction of errors more difficult, as there are no clear problems that can be resolved easily by identifying a cue the learner does not pick up on. The current scores are most likely limited by the small size (339 instances) and limited coverage of the dataset, and we expect to see improvements once more data has been gathered.

4.3 The Importance of Feature Subset Selection

A brief look at the top performing feature sets in Table 10.5 (repeated from Table 10.2) tells us that there isn't simply one set of features that is best for all the

[7]The statistical significance of the difference between TiMBL's results and the bag-of-words baseline was tested using a paired t-test which yielded a p-value of 0.0257, which is considered statistically significant.

	features	# of feats	filter metric	accuracy
TiMBL	w1, l3, d	1000	Gini	52.5%
Naïve Bayes	l1, w3, wls	all	n/a	46.6%
SVM	c1, l3, wlc	250	infogain	35.7%
CRF	w1, wlc	500	infogain	47.8%

Table 10.5: Top results per learner

different learners, as each learner performs best with different types of features. Classifiers also seem to react differently to feature subset selection, with Naïve Bayes performing best when provided with all features while other learners benefit from using a reduced set of features. To illustrate the importance of finding the right feature set for the right learner, Table 10.6 shows how Naïve Bayes, SVM and CRF fare when we train them using the top feature set for TiMBL. There is a clear and significant drop in performance when using a feature set that isn't adapted for the learner in use.

	Naïve Bayes	SVM	CRF
top accuracy	46.6%	35.7%	47.8%
using TiMBL features	30.4%	26.5%	36.9%
difference	-16.2%	-9.2%	-10.9%

Table 10.6: Importance of adapted feature subsets per learner

While different feature subsets are needed to get the best out of each learner, there are some elements that stay constant. Word tokens and lemmas, for instance, are consistently present in the top feature combinations for every learner. Table 10.7 illustrates this by comparing the top feature combination for each learner to the first combination *not* using word tokens or lemmas. There is only a small performance drop for Naïve Bayes, SVMs and CRF, but TiMBL results without words or lemmas are significantly lower.

	TiMBL	Naïve Bayes	SVM	CRF
top accuracy	52.5%	46.6%	35.7%	47.8%
without tokens or lemmas	38.4%	41.9%	33.9%	46.6%
difference	-14.1%	-4.7%	-2.2%	-1.2%

Table 10.7: Importance of word tokens and lemmas as features

When we examine the specific words and lemmas that the filter metrics propose as the most relevant features, we see that most of them seem instinctively plausible as important cues related to Leary's Rose. Question marks and exclamation marks, for instance, are amongst the 10 most relevant features, and their relevance is easily

illustrated by examining the following sentences from the *deLearyous* dataset:

- Wat vindt u zelf van dit voorstel**?** (*What do you think of this suggestion?*) - **TOG_B**
- Zoek het zelf maar uit**!** (*Just figure it out for yourself!*) - **OPP_A**

In the first sentence, the speaker positions himself in an inferior, expectant position towards the listener. The listener is given full control over the situation as he can still reject the speaker's suggestion. The fact that the sentence is a question contributes to the speaker's submissive, cautious position. The question mark is therefore associated with the "below" half of Leary's Rose.

In the second sentence, the exclamation mark accentuates the dominant position of the speaker. The speaker is annoyed or angry at the listener and authoritatively tells him what to do. The exclamation mark is thus associated with the "above" half of the Rose, since it usually points to dominant behavior.

It should be noted that exclamation marks needn't always express dominance, and question marks don't always point to submission. These features on their own are far from enough to classify sentences into the quadrants of Leary's Rose. It's only in combination with many other features that they turn out to be especially useful.

Another interesting feature pair that shows up in the top 10 most relevant features is the distinction between the personal pronouns "u" and "je". "U" is a Dutch pronoun that marks politeness, while "je" is the more general unmarked second person pronoun. Using one or the other tells us a lot about the power dynamics in a conversation:

- Ik begrijp dat **u** kwaad bent, mevrouw... (*I understand that you're angry, Mrs...*) - **TOG_B**
- Ik verwacht dat **je** naar me luistert! (*I expect you to listen to me!*) - **OPP_A**

Finally, there's "we", the first person plural pronoun, which instinctively guides the interpretation of a sentence to the "together" half of Leary's Rose:

- Gaat u even zitten, dan zoeken **we** samen naar een oplossing. (*"Please sit down, we'll find an answer together."*) - **TOG_A**

4.4 The Importance of Parameter Optimization

For those learners that allow different parameters, determining the most efficient parameters may be just as important as choosing the appropriate feature types. Indeed, a classifier that has been trained with suboptimal features but uses good parameters might not be far off, performance-wise, from a learner trained with the best features but using the "wrong" parameters (see also Daelemans et al. (2003)). Table 10.8 illustrates this problem: it compares the results of a classification task with a suboptimal set of features (but with "good" learner parameters) with the results of a classification task where the learner parameters have not been optimized (but the features are the "correct" ones).

For the purpose of this experiment, the "wrong" feature types used were character bigrams, part-of-speech trigrams and unigrams of syntactic functions. The "bad" parameters were the default parameters (for TiMBL: k = 1, weighting = gainratio, distance metric = weighted overlap; for SVMs: training error/margin trade-off = 0.01).

	TiMBL	SVM
top accuracy ("good" features and parameters)	52.5%	35.7%
using "bad" features	**25.0%**	26.3%
using "bad" parameters	30.1%	**25.1%**

Table 10.8: Performance when using "bad" learner parameters or feature subsets

For the TiMBL experiments, the results of the classification task using "bad" parameteres are almost as low as for the experiment using the "wrong" feature types. For SVMs, the comparison is even more striking, as the optimized learner trained on the "bad" features even outperforms (though not significantly so) the learner trained on the "good" ones without parameter optimization.

5 Conclusion

We have outlined the natural language component of a new project, project *deLearyous*, which aims to create a serious video game that will allow players to interact with a virtual character using Dutch natural language sentences. Through this interaction with the virtual character, the players should be able to improve their communication skills by learning to use Leary's Rose, a framework for interpersonal communication. The natural language processing component is of primordial importance to *deLearyous*, and we described how we attempted to identify the position of the player in Leary's Rose on the basis of their textual input.

We chose to use machine learning techniques to perform this classification task, and we used a small dataset of 339 Dutch sentences to test four different learner algorithms: a memory-based learner (TiMBL), a Naïve Bayes classifier, a multi-class implementation of Support Vector Machines and Conditional Random Fields. So far, we managed to achieve an accuracy of 52.5% using TiMBL and a combination of word token unigrams, lemma trigrams and average word length as features.

We have determined that word tokens and lemmas are important feature types for all learners, and have looked at some of these top features in more detail to determine how they relate to Leary's Rose. We have also shown the influence that the use different feature subsets has on each learner, and noted that finding the right set of learner parameters is at least as important as finding a good feature subset.

6 Future Research

At the moment, we can only classify one out of two sentences into the correct quadrant of Leary's Rose, which is insufficient if *deLearyous* is to be a useful tool for communication training. An additional problem is that the final product will not restrict itself to the four quadrants on the Rose, but it will incorporate dynamics relating to the eight octants, which complicates the classification task significantly.

There are several elements that will help balance out these difficulties, however. Right now, the learners have been trained on a very small dataset that only sparsely covers a wide array of different possible communication scenarios. As the *deLearyous* project moves forward, one specific scenario will be chosen, which will allow us to expand the dataset with relevant instances. Additionally, each sentence is now looked at in isolation, i.e. the machine learner has no idea of what has happened in the conversation prior to the current sentence. Once the scenario has been established, however, it will be possible to also integrate information about the context, hopefully improving classification accuracy in the progress.

Until now, we have not used any features based on emotion or sentiment lexicons. Most existing resources (SentiWordNet, WordNet Affect, etc.) have been compiled for the English language only, though Jijkoun and Hofmann (2009) have sucessfully constructed a Dutch subjectivity lexicon based on the English lexicon of OpinionFinder (Wilson et al. 2005) and Cornetto (Vossen et al. 2007) –an extension of the Dutch WordNet. Since integrating sentiment orientation features into the machine-learning approach has been successful for English (Mishne (2005), Keshtkar and Inkpen (2009), Inkpen et al. (2009)), there is reason to believe that this technique will also prove to be useful for Dutch.

A final aspect that needs to be researched is the problem of reliability. Until now, all data was manually labeled by one annotator, but identifying emotions in written text is not straightforward, even for humans. It is therefore important to have several annotators label the data and to check inter- and intra-annotator agreement scores. On the basis of these scores, we can then determine a ceiling beyond which we can not realistically expect the machine learners to perform.

7 Acknowledgements

This study was made possible through financial support from the IWT (the government agency for Innovation by Science and Technology, TETRA-project *deLearyous*). We would also like to thank the CLIN reviewers for their helpful feedback.

References

Balahur, Alexandra, Ralf Steinberger, Mijail Alexandrov Kabadjov, Vanni Zavarella, Erik Van der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva (2010), Sentiment analysis in the news, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Crammer, Koby and Yoram Singer (2002), On the algorithmic implementation of multiclass kernel-based vector machines, *Journal of Machine Learning Research* pp. 265–292, MIT Press.

Daelemans, Walter and Antal van den Bosch (2005), *Memory-Based Language Processing*, Cambridge, UK: Cambridge University Press.

Daelemans, Walter, Véronique Hoste, Fien De Meulder, and Bart Naudts (2003), Combined optimization of feature selection and algorithm parameter interaction in machine learning of language, *Proceedings of the 14th European Conference on Machine Learning (ECML-2003)*, Berlin: Springer, pp. 84–95.

Goyal, Amit, Ellen Riloff, Hal Daumé III, and Nathan Gilbert (2010), Toward plot units: Automatic affect state analysis, *Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*.

Inkpen, Diana, Fazel Keshtkar, and Diman Ghazi (2009), Analysis and generation of emotion in texts, *in* Clujeana, Presa Universitara, editor, *KEPT 2009 Knowledge Engineering-Principles and Techniques, Selected Papers*, Militon Frentiu and Horia F. Pop, pp. 3–13.

Jijkoun, Valentin and Katja Hofmann (2009), Generating a non-english subjectivity lexicon: Relations that matter., *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, The Association for Computer Linguistics, pp. 398–405.

Keshtkar, Fazel and Diana Inkpen (2009), Using sentiment orientation features for mood classification in blogs, *IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE 2009), Dalian, China*.

Leary, T. (1957), *Interpersonal Diagnosis of Personality: Functional Theory and Methodology for Personality Evaluation*, New York: Ronald Press.

Luyckx, Kim and Walter Daelemans (2005), Shallow text analysis and machine learning for authorship attribution, *Computational Linguistics in the Netherlands 2004: Selected papers from the Fifteenth CLIN Meeting*, pp. 149–160.

Mishne, Gilad (2005), Experiments with mood classification in blog posts, *Proceedings of the 1st Workshop on Stylistic Analysis Of Text For Information Access*.

Okazaki, Naoaki (2007), CRFsuite: a fast implementation of Conditional Random Fields (CRFs). http://www.chokkan.org/software/crfsuite/.

Rentoumi, Vassiliki, Stefanos Petrakis, Manfred Klenner, George A. Vouros, and Vangelis Karkaletsis (2010), United we stand: Improving sentiment analysis by joining machine learning and rule based methods, *in* Calzolari, Nicoletta, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, European Language Resources Association (ELRA), Valletta, Malta.

Sebastiani, Fabrizio (2002), Machine learning in automated text categorization,

ACM Comput. Surv. **34** (1), pp. 1–47, ACM, New York, NY, USA.

Tsur, Oren, Dmitry Davidov, and Ari Rappoport (2010), ICWSM - a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews, *in* Hearst, Marti, William Cohen, and Samuel Gosling, editors, *Proceedings of the Fourth International Conference on Weblogs and Social Media (ICWSM-2010)*, The AAAI Press, Menlo Park, California.

van den Bosch, Antal, Bertjan Busser, Sander Canisius, and Walter Daelemans (2007), An efficient memory-based morphosyntactic tagger and parser for dutch, *in* Eynde, F. Van, P. Dirix, I. Schuurman, and V. Vandeghinste, editors, *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting*, Leuven, Belgium, pp. 99–114. http://ilk.uvt.nl/tadpole/.

van Dijk, Bert (2000), *Beïnvloed anderen, begin bij jezelf. Over gedrag en de Roos van Leary*, 4th ed., Thema.

Vossen, Piek, Katja Hofmann, Maarten de Rijke, Erik Tjong Kim Sang, and Koen Deschacht (2007), The Cornetto database: Architecture and user-scenarios, *in* Moens, M.-F., T. Tuytelaars, and A.P. de Vries, editors, *Proceedings DIR 2007*, pp. 89–96.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann (2005), Recognizing contextual polarity in phrase-level sentiment analysis, *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pp. 347–354.

Paraphrasing Headlines by Machine Translation

Sentential paraphrase acquisition and generation using Google News

Sander Wubben, Antal van den Bosch, and Emiel Krahmer

Tilburg Centre for Cognition and Communication
Tilburg University

Abstract

In this paper we investigate the automatic collection, generation and evaluation of sentential paraphrases. Valuable sources of paraphrases are news article headlines; they tend to describe the same event in various different ways, and can easily be obtained from the web. We describe a method for generating paraphrases by using a large aligned monolingual corpus of news headlines acquired automatically from Google News and a standard Phrase-Based Machine Translation (PBMT) framework. The output of this system is compared to a word substitution baseline. Human judges prefer the PBMT paraphrasing system over the word substitution system. We compare human judgements to automatic judgement measures and demonstrate that the BLEU metric correlates well with human judgements provided that the generated paraphrase is sufficiently different from the source sentence.

1 Introduction

Text-to-text generation is an increasingly studied subfield in natural language processing. In contrast with the typical natural language generation paradigm of converting concepts to text, in text-to-text generation a source text is converted into a target text that approximates the meaning of the source text. Text-to-text generation extends to such varied tasks as summarization (Knight and Marcu 2002), question-answering (Lin and Pantel 2001), Machine Translation, and paraphrase generation.

For text-to-text generation it is important to know which words and phrases are semantically close or exchangable in which contexts. While there are various resources available that capture such knowledge at the word level (e.g., synonymic knowledge in WordNet), this kind of information is much harder to get by at the phrase or even at the sentence level. The paraphrasing task extends from the word level up to the discourse level; a WordNet-like resource at the paraphrase level would be needed to generate paraphrases of new, unseen text. Therefore, paraphrase acquisition can be considered an important technology for producing resources for text-to-text generation. Paraphrase generation has already proven to be valuable for Question Answering (Lin and Pantel 2001, Riezler et al. 2007), Machine Translation (Callison-Burch et al. 2006) and the evaluation thereof (Russo-Lassner et al. 2006, Kauchak and Barzilay 2006, Zhou et al. 2006), but also for text simplification and explanation.

Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

Paraphrase generation is the process of transforming a source sentence into a target sentence in the same language which differs in form from the source sentence, but approximates its meaning. Paraphrasing is often used as a subtask in more complex NLP applications to allow for more variation in text strings presented as input, for example to generate paraphrases of questions that in their original form cannot be answered (Lin and Pantel 2001, Riezler et al. 2007), or to generate paraphrases of sentences that failed to translate (Callison-Burch et al. 2006). Paraphrasing has also been used in the evaluation of Machine Translation system output (Russo-Lassner et al. 2006, Kauchak and Barzilay 2006, Zhou et al. 2006). Adding certain constraints to paraphrasing allows for additional useful applications. When the constraint is specified that a paraphrase should be shorter than the input text, paraphrasing can be used for sentence compression (Knight and Marcu 2002, Barzilay and Lee 2003). Another specific task that can be approached this way is text simplification for question answering or subtitle generation (Daelemans et al. 2004).

In this paper we regard the generation of sentential paraphrases as a monolingual Machine Translation task, where the source and target languages are the same (Quirk et al. 2004). However, there are two problems that have to be dealt with to make this approach work, namely obtaining a sufficient amount of examples, and a proper evaluation methodology. As (Callison-Burch et al. 2008) argue, automatic evaluation of paraphrasing is problematic. The essence of paraphrasing is to generate a sentence that is structurally different from the source. Automatic evaluation metrics in related fields such as standard multilingual Machine Translation operate on a notion of similarity, while paraphrasing also centers around achieving dissimilarity. Besides the evaluation issue, another problem is that for an data-driven Machine Translation account of paraphrasing to work, a large collection of data is required. In this case, this would have to be pairs of sentences that are paraphrases of each other. So far, paraphrasing data sets of sufficient size have been mostly lacking. The work on paraphrasing has also mainly been focused on phrases as opposed to sentences. We argue that the headlines aggregated by Google News offer an attractive avenue.

2 Data Collection

Currently few resources are available for paraphrasing; one example is the Microsoft Paraphrase Corpus (MSR) (Dolan et al. 2004, Nelken and Shieber 2006), which is relatively small: it contains 139,000 aligned paraphrases. In this study we explore the use of a large, automatically acquired aligned paraphrase corpus. Where previous work has focused on aligning news-items at the paragraph and sentence level (Barzilay and Elhadad 2003), we limit ourselves to aligning headlines of news articles. We think this approach will enable us to harvest reliable training material for paraphrase generation fast and efficiently, without having to worry too much about the problems that arise when trying to align complete news articles. Google News is such a vast resource that we already get a lot of data by looking at headlines alone.

For the development of our system we use data which was obtained in the DAESO-project. This project is an ongoing effort to build a Parallel Monolingual Treebank for Dutch (Marsi and Krahmer 2007) and will be made available through the Dutch HLT Agency. Part of the data in the DAESO-corpus consists of headline clusters crawled from Google News in the period April–August 2006. Google News uses clustering algorithms that consider the full text of each news article, as well as other features such as temporal and category cues, to produce sets of articles related topically. The crawler stored the headline and the first 150 characters of the article of each news article crawled from the Google News website. Roughly 13,000 Dutch clusters were retrieved, 450 MB in size. Table 11.1 shows part of a cluster. It is clear that although clusters deal roughly with one subject, the headlines can represent quite a different perspective on the content of the article. To obtain only paraphrase pairs, the clusters need to be more coherent. To that end, in the DAESO project 865 clusters were manually subdivided into sub-clusters of headlines that show clear semantic overlap. Sub-clustering is no trivial task, however. Some sentences are very clearly paraphrases, but consider for instance the sentences in the example containing 'Afghanistan' or 'Uruzgan'. they can be seen as paraphrases of each other, but then the reader must know that 'Uruzgan' is a province in Afghanistan where the Dutch mission is stationed. Also, there are numerous headlines that can not be sub-clustered, such as the first three headlines shown in the example.

This annotated data is used to develop a method of automatically obtaining paraphrase pairs from headline clusters. We divide the annotated headline clusters in a development set of 40 clusters, while the remainder is used as test data. The headlines are stemmed using the porter stemmer for Dutch (Kraaij and Pohlmann 1994) Instead of a word overlap measure as used by Barzilay and Elhadad (Barzilay and Elhadad 2003), we use a modified $TF * IDF$ word score as was suggested by Nelken and Shoeber (Nelken and Shieber 2006). Each sentence is viewed as a document, and each original cluster as a collection of documents. For each stemmed word i in sentence j, $TF_{i,j}$ is a binary variable indicating if the word occurs in the sentence or not. The $TF * IDF$ score can then be stated as follows:

$$\text{TF.IDF}_\text{i} = TF_{i,j} \cdot \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

$|D|$ is the total number of sentences in the cluster and $|\{d_j : t_i \in d_j\}|$ is the number of sentences that contain the term t_i. These scores are used in a vector space representation. The similarity between headlines can be calculated by using a similarity function on the headline vectors, such as Cosine similarity.

2.1 Clustering

Our first approach is to use a clustering algorithm to cluster similar headlines. The original Google News headline clusters are reclustered into finer grained subclusters. We use the k-means implementation in the CLUTO[1] software package.

[1] `http://glaros.dtc.umn.edu/gkhome/views/cluto/`

Kamp : Veiligheid grootste probleem in Uruzgan *(Kamp: Security biggest problem in Uruzgan)*
Met gevechtsheli op Afghaanse theevisite *(With attack helicopter on Afghan tea-visit)*
Bevel overgedragen aan Nederlandse commandant *(Command transferred to Dutch commander)*
Nederlandse missie Uruzgan officieel begonnen *(Dutch mission Uruzgan officially started)* Nederlandse opbouwmissie in Afghanistan begint *(Dutch construction mission in Afghanistan begins)* Missie Uruzgan begonnen *(Mission Uruzgan had begun*
Soldaten opbouwmissie Uruzgan keren terug *(Soldiers construction mission return)* Eerste militairen komen terug uit Afghanistan *(First servicemen come back from Afghanistan)* Eerste groep militairen Afghanistan keert terug *(First group of servicemen return from Afghanistan)* Kwartiermakers keren terug uit Uruzgan *(Quartermasters return from Uruzgan)*
Opgelucht onthaal van militairen uit Uruzgan *(Relieved welcome of servicemen from Uruzgan)* Opgelucht onthaal van Uruzgan-gangers *(Relieved welcome of Uruzgan-goers)*

Table 11.1: Part of a sample headline cluster crawled in August 2006. The original headlines are displayed as they were clustered by the annotators.

The k-means algorithm is an algorithm that assigns k centers to represent the clustering of n points ($k < n$) in a vector space. The total intra-cluster variances is minimized by the function

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

where μ_i is the centroid of all the points $x_j \in S_i$.

The PK1 cluster-stopping algorithm as proposed by Pedersen and Kulkarni (Pedersen and Kulkarni 2006) is used to find the optimal k for each sub-cluster:

$$PK1(k) = \frac{Cr(k) - mean(Cr[1...deltaK])}{std(Cr[1...deltaK])}$$

Here, Cr is a criterion function. As soon as $PK1(k)$ exceeds a threshold, $k-1$ is selected as the optimum number of clusters.

To find the optimal threshold value for cluster stopping, optimization is performed on the development data. Our optimization function is an F-score:

$$F_\beta = \frac{(1+\beta^2) \cdot (\text{precision} \cdot \text{recall})}{(\beta^2 \cdot \text{precision} + \text{recall})}$$

We evaluate the number of aligments between possible paraphrases. For instance, in a cluster of four sentences, $\binom{4}{2} = 6$ alignments can be made. In our case, precision is the number of alignments retrieved from the clusters which are relevant, divided by the total number of retrieved alignments. Recall is the number of relevant retrieved alignments divided by the total number of relevant alignments.

We use an F_β-score with a β of 0.25 as we favor precision above recall. We do not want to optimize on precision alone, because we still want to retrieve a fair amount of paraphrases and not only the ones that are very similar. Through optimization on our development set, we find an optimal threshold for the PK1 algorithm $th_{pk1} = 1$. For each original cluster, k-means clustering is then performed using the k found by the cluster stopping function. In each newly obtained cluster all headlines can be aligned with each other.

2.2 Pairwise similarity

Our second approach is to directly calculate similarities for each pair of headlines within a cluster. If the similarity exceeds a certain threshold, the pair is accepted as a paraphrase pair. If it is below the threshold, it is rejected. However, as Barzilay and Elhadad (Barzilay and Elhadad 2003) have pointed out, sentence mapping in this way is only effective to a certain extent. Beyond that point, context is needed. With this in mind, we adopt two thresholds and the Cosine similarity function to calculate the similarity between two sentences:

$$cos(\theta) = \frac{V1 \cdot V2}{||V1||\ \ ||V2||}$$

Type	Precision	Recall
k-means clustering clusters only	0.91	0.43
k-means clustering all headlines	0.66	0.44
pairwise similarity all headlines	0.76	0.41

Table 11.2: Precision and Recall for both methods

where $V1$ and $V2$ are the vectors of the two sentences being compared. If the similarity is higher than the upper threshold, it is accepted. If it is lower than the lower theshold, it is rejected. In the remaining case of a similarity between the two thresholds, similarity is calculated over the contexts of the two headlines, namely the text snippet that was retrieved with the headline. If this similarity exceeds the upper threshold, it is accepted. Threshold values as found by optimizing on the development data using again an $F_{0.25}$-score, are $Th_{lower} = 0.2$ and $Th_{upper} = 0.5$. An optional final step is to add alignments that are implied by previous alignments. For instance, if headline A is paired with headline B, and headline B is aligned to headline C, headline A can be aligned to C as well. We do not add these alignments, because particularly in large clusters when one wrong alignment is made, this process adds a large amount of incorrect alignments.

2.3 Results

The 825 clusters in the test set contain 1,751 sub-clusters in total. In these sub-clusters, there are 6,685 clustered headlines. Another 3,123 headlines remain unclustered. Table 11.2 displays the paraphrase detection precision and recall of our two approaches. It is clear that k-means clustering performs well when all unclustered headlines are artificially ignored. In the more realistic case when there are also items that cannot be clustered, the pairwise calculation of similarity with a back off strategy of using context performs better when we aim for higher precision.

2.4 Obtaining headline paraphrase pairs

We choose the pairwise similarity approach to extract paraphrasing headline pairs from our much larger extracted English dataset, consisting of roughly 30,000 English headlines that appeared in Google News over the period of April to September 2006, 3 GB in size. Using this method we end up with a collection of 7,400,144 pairwise alignments of 1,025,605 unique headlines[2]. An example of alignments created with this approach is in Figure 1,

[2]This list of aligned pairs will be made available online.

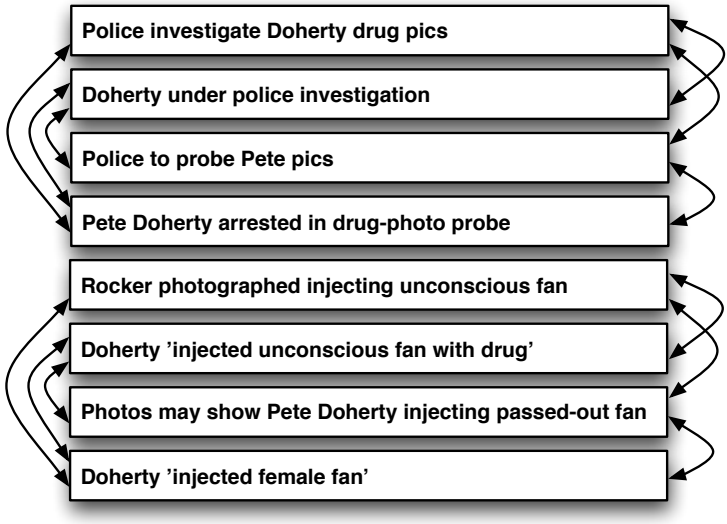


Figure 1: Part of a sample headline cluster, with aligned paraphrases

3 Paraphrase Generation

In our approach we use the collection of automatically obtained aligned headlines to train a paraphrase generation model using a Phrase-Based Machine Translation (PBMT) framework. We compare this approach to a word substitution baseline. The generated paraphrases along with their source headlines are presented to human judges, whose ratings are compared to the BLEU (Papineni et al. 2002), METEOR (Banerjee and Lavie 2005) and ROUGE (Lin 2004) automatic evaluation metrics.

3.1 Phrase-Based MT

We use the MOSES package to train a PBMT model (Koehn et al. 2007). Such a statistical model normally finds a best translation $\tilde{e}$ of a text in language f to a text in language e by combining a translation model $p(f|e)$ with a language model $p(e)$:

$$\tilde{e} = arg \max_{e \in e^*} p(f|e)p(e)$$

GIZA++ is used to perform the word alignments, which are later combined into phrase alignments in the MOSES pipeline (Och and Ney 2003) to build the paraphrase model. We first tokenize our data before training a recaser. We then lowercase all data and use all unique headlines in the training data to train a language model with the SRILM toolkit (Stolcke 2002). Then we invoke the GIZA++

System	Headline
Source	Florida executes notorious serial killer
PBMT	Serial killer executed in Florida
Word Sub.	Florida executes ill-famed series slayer
Source	Dublin evacuates airport due to bomb scare
PBMT	Dublin airport evacuated after bomb threat
Word Sub.	Dublin evacuates airdrome due to bomb panic
Source	N. Korea blasts nuclear sanctions
PBMT	N. Korea nuclear blast of sanctions
Word Sub.	N. Korea blasts atomic sanctions
Source	Israeli raid in Lebanon kills 54
PBMT	Israeli raid kills 54 in Lebanon
Word Sub.	Israeli foray in Lebanon kills 54

Table 11.3: Examples of generated paraphrases

aligner using the 7M training paraphrase pairs. We run GIZA++ with standard settings and we perform no optimization. Finally, we use the MOSES decoder to generate paraphrases for our test data.

Instead of assigning equal weights to language and translation model, we assign a larger weight of 0.7 to the language model to generate better formed (but more conservative) paraphrases. Because dissimilarity is a factor that is very important for paraphrasing but not implemented in a PBMT model, we perform post-hoc reranking based on dissimilarity. We clearly want our output to be different from our input after all. For each headline in the testset we generate the ten best paraphrases as scored by the decoder and then rerank them according to dissimilarity to the source using the Levenshtein distance measure modified to the word level. This means we look at insertion, deletion and substitution of words. The resulting headlines are recased using the previously trained recaser.

3.2 Word Substitution

The PBMT results are compared with a simple word substitution baseline. For each noun, adjective and verb in the sentence this model takes that word and its Part of Speech tag and retrieves from WordNet its most frequent synonym from the most frequent synset containing the input word. If no relevant alternative is found, the word is left unaltered. We use the Memory Based Tagger (Daelemans et al. 1996) trained on the Brown corpus to generate the POS-tags. The WordNet::QueryData[3] Perl module is used to query WordNet (Fellbaum 1998). Generated headlines and their source for both systems are given in Table 11.3.

[3] http://search.cpan.org/dist/WordNet-QueryData/QueryData.pm

operation	sentences
single word replacement	80
word deletion or insertion	55
word/phrase reordering	18
phrase replacements	60
sentence rewriting	3

Table 11.4: Analysis of the generated paraphrases by the PBMT system indicating the number of sentences containing one or more of the specified edit operation.

4 Evaluation

A human judgement study was set up to evaluate the generated paraphrases, and the human judges' ratings are compared to automatic evaluation measures in order to gain more insight in the automatic evaluation of paraphrasing.

4.1 Method

We randomly select 160 headlines from all headlines that meet the following criteria: the headline has to be comprehensible without reading the corresponding news article, both systems have to be able to produce a paraphrase for each headline, and there have to be a minimum of eight paraphrases for each headline. We need these paraphrases as multiple references for our automatic evaluation measures to account for the diversity in real-world paraphrases, as the aligned paraphrased headlines in Figure 1 witness.

The judges are presented with the 160 headlines, along with the paraphrases generated by both systems. The order of the headlines is randomized, and the order of the two paraphrases for each headline is also randomized to prevent a bias towards one of the paraphrases. The judges are asked to rate the paraphrases on a 1 to 7 scale, where 1 means that the paraphrase is very bad and 7 means that the paraphrase is very good. The judges were instructed to base their overall quality judgement on whether the meaning was retained, the paraphrase was grammatical and fluent, and whether the paraphrase was in fact different from the source sentence. Ten judges rated two paraphrases per headline, resulting in a total of 3,200 scores. All judges were blind to the purpose of the evaluation and had no background in paraphrasing research.

system	mean	stdev.
PBMT	4.60	0.44
Word Substitution	3.59	0.64

Table 11.5: Results of human judgements (N = 10)

System	BLEU	ROUGE-1	ROUGE-2	ROUGE-SU4	METEOR	Lev.dist.	Lev. stdev.
PBMT	0.51	0.76	0.36	0.42	0.71	2.76	1.35
Wordsub.	0.25	0.59	0.22	0.26	0.54	2.67	1.50
Source	0.61	0.80	0.45	0.47	0.77	0	0

Table 11.6: Automatic evaluation and sentence Levenshtein scores

4.2 Results

The average scores assigned by the human judges to the output of the two systems are displayed in Table 11.5. These results show that the judges rated the quality of the PBMT paraphrases significantly higher than those generated by the word substitution system ($t(18) = 4.11, p < .001$).

Results from the automatic measures as well as the Levenshtein distance are listed in Table 11.6. We use a Levenshtein distance over tokens instead of characters. First, we observe that both systems perform roughly the same amount of edit operations on a sentence, resulting in a Levenshtein distance over words of 2.76 for the PBMT system and 2.67 for the Word Substitution system. BLEU, METEOR and three typical ROUGE metrics[4] all rate the PBMT system higher than the Word Substitution system. Notice also that the all metrics assign the highest scores to the original sentences, as is to be expected: because every operation we perform is in the same language, the source sentence is also a paraphrase of the reference sentences that we use for scoring our generated headline. If we pick a random sentence from the reference set and score it against the rest of the set, we obtain similar scores. This means that this score can be regarded as an upper bound score for paraphrasing. However, this also shows that these measures cannot be used directly as an automatic evaluation method of paraphrasing, as they assign the highest score to the "paraphrase" in which nothing has changed. The scores observed in Table 11.6 do indicate that the paraphrases generated by PBMT are less well formed than the original source sentence.

Table 11.4 shows a breakdown of the paraphrasing operations the PBMT approach has performed. The number indicates the amount of sentences out of the 160 that contain the specific edit operation. Phrase replacements should be interpreted as a replacement operating involving multi-word expressions. Sentence rewriting means that the sentence is fundamentally changed in its entirety, for instance changing from passive to active and vice versa. The first two sentences in Table 11.3 are examples of this.

There is an overall medium correlation between the BLEU measure and human judgements ($r = 0.41, p < 0.001$). We see a lower correlation between the various ROUGE scores and human judgements, with ROUGE-1 showing the highest

[4]ROUGE-1, ROUGE-2 and ROUGE-SU4 are also adopted for the DUC 2007 evaluation campaign, `http://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html`

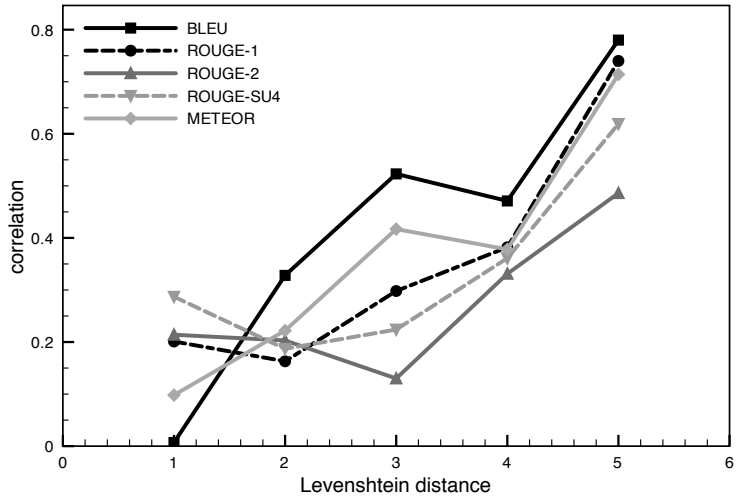


Figure 2: Correlations between human judgements and automatic evaluation metrics for various edit distances

correlation ($r = 0.29, p < 0.001$). Between the two lies the METEOR correlation ($r = 0.35, p < 0.001$). However, if we split the data according to Levenshtein distance, we observe that we generally get a higher correlation for all the tested metrics when the Levenshtein distance is higher, as visualized in Figure 2. At Levenshtein distance 5, the BLEU score achieves a correlation of 0.78 with human judgements, while ROUGE-1 manages to achieve a 0.74 correlation. Beyond edit distance 5, data sparsity occurs.

5 Conclusion

In this paper we have shown that with an automatically obtained parallel monolingual corpus with several millions of paired examples, it is possible to develop a sentential paraphrase generation system based on a PBMT framework. We have described a method to align headlines extracted from Google News based on similarity between the two headlines. We have shown that a Cosine similarity function comparing headlines and using a back off strategy to compare context can be used to extract Dutch paraphrase pairs at a precision of 0.76. Although we could aim for a higher precision by assigning higher values to the thresholds, we still want to retain some recall and variation in our paraphrases.

The use of a PBMT framework to exploit this resource of aligned headlines is a feasible strategy; human judges preferred the output of our PBMT system over the output of a simple word substitution system. We have also addressed the

problem of automatic paraphrase evaluation. We measured BLEU, METEOR and ROUGE scores, and observed that these automatic scores correlate with human judgements to some degree, but that the correlation is highly dependent on edit distance. At low edit distances automatic metrics fail to properly assess the quality of paraphrases, whereas at edit distance 5 the correlation of BLEU with human judgements is 0.78, indicating that at higher edit distances these automatic measures can be utilized to rate the quality of the generated paraphrases. From edit distance 2, BLEU correlates best with human judgements, which suggests that Machine Translation evaluation metrics might be better for automatic paraphrase evaluation than summarization metrics.

6 Discussion and future work

The data we used for paraphrasing consists of headlines. Of course headlines use a special kind of language. In headlines articles and most forms of the verb 'to be' are often omitted. Most headlines are in simple present tense and written in telegraphic style and use a lot of abbreviations and metonyms to denote companies and organizations (i.e. 'Wall Street'). This means that the paraphrase patterns we learn are those used in headlines and possibly different from normal conversational language. The advantage of our approach is however that it paraphrases those parts of sentences that it can paraphrase, and leaves those parts that are unknown intact. This is different when we perform standard multilingual translation: if the unknown word is not a proper noun, it can not be left untranslated. It is straightforward to train a language model on in-domain text and use the translation model acquired from the headlines to generate paraphrases for other domains. We are of course also interested in capturing paraphrase patterns existing in other domains, but acquiring parallel paraphrase corpora for different domains is no trivial task.

Our plans for future work are twofold: on the one hand we wish to improve the automatic paraphrase generation process by augmenting the phrase alignment phase, using linguistic information in addition to the statistical models that are employed by GIZA++. We think that due to the monolingual nature of paraphrasing, linguistic information can be used with great effect. In addition, we plan to investigate if our paraphrase generation approach is applicable to comparable fields such as sentence compression and sentence simplification. Our goal is to develop a proper uniform paraphrasing model that is able to take into account different constraints we want to impose on our paraphrases during the decoding process. Such constraints can be dissimilarity for normal paraphrasing, but also simplicity or readability in the case of paraphrasing for sentence simplification and length for sentence compression. These constraints can be taken into account by adding scores for these constraints in addition to scores as provided by the language and translation model. On the topic of automatic evaluation, we aim to define an automatic paraphrase generation assessment score. An automatic paraphrase evaluation measure should be able to recognize that a good paraphrase is a well-formed sentence in the source language, yet at the same time it is clearly dissimilar to the source.

References

Banerjee, Satanjeev and Alon Lavie (2005), METEOR: An automatic metric for MT evaluation with improved correlation with human judgments, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Association for Computational Linguistics, Ann Arbor, Michigan, pp. 65–72. http://www.aclweb.org/anthology/W/W05/W05-0909.

Barzilay, Regina and Lillian Lee (2003), Learning to paraphrase: an unsupervised approach using multiple-sequence alignment, *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 16–23. http://portal.acm.org/citation.cfm?id=1073445.1073448.

Barzilay, Regina and Noemie Elhadad (2003), Sentence alignment for monolingual comparable corpora, *Proceedings of the 2003 conference on Empirical methods in natural language processing*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 25–32.

Callison-Burch, Chris, Philipp Koehn, and Miles Osborne (2006), Improved statistical machine translation using paraphrases, *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 17–24.

Callison-Burch, Chris, Trevor Cohn, and Mirella Lapata (2008), Parametric: an automatic evaluation metric for paraphrasing, *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 97–104.

Daelemans, Walter, Anja Hothker, and Erik Tjong Kim Sang (2004), Automatic sentence simplification for subtitling in dutch and english., *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pp. 1045–1048. http://www.cnts.ua.ac.be/Publications/2004/DHT04.

Daelemans, Walter, Jakub Zavrel, Peter Berck, and Steven Gillis (1996), MBT: A Memory-Based Part of Speech Tagger-Generator, *Proc. of Fourth Workshop on Very Large Corpora*, ACL SIGDAT, pp. 14–27.

Dolan, Bill, Chris Quirk, and Chris Brockett (2004), Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources, *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, p. 350.

Fellbaum, Christiane, editor (1998), *WordNet: An Electronic Lexical Database*, The MIT Press. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/026206197X.

Kauchak, David and Regina Barzilay (2006), Paraphrasing for automatic evaluation, *Proceedings of the Human Language Technol-*

ogy Conference of the NAACL, Main Conference, Association for Computational Linguistics, New York City, USA, pp. 455–462. http://www.aclweb.org/anthology/N/N06/N06-1058.

Knight, Kevin and Daniel Marcu (2002), Summarization beyond sentence extraction: a probabilistic approach to sentence compression, *Artif. Intell.* **139** (1), pp. 91–107, Elsevier Science Publishers Ltd., Essex, UK.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris C. Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst (2007), Moses: Open source toolkit for statistical machine translation, *ACL*, The Association for Computer Linguistics. http://dblp.uni-trier.de/rec/bibtex/conf/acl/KoehnHBCFBCSMZDBCH07.

Kraaij, Wessel and Renée Pohlmann (1994), Porter's stemming algorithm for Dutch, *Informatiewetenschap 1994: Wetenschappelijke bijdragen aan de derde STINFON Conferentie*, pp. 167–180.

Lin, Chin-Yew (2004), Rouge: A package for automatic evaluation of summaries, *Proc. ACL workshop on Text Summarization Branches Out*, p. 10. http://research.microsoft.com/ cyl/download/papers/WAS2004.pdf.

Lin, Dekang and Patrick Pantel (2001), DIRT: Discovery of inference rules from text, *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, New York, NY, USA, pp. 323–328.

Marsi, Erwin and Emiel Krahmer (2007), Annotating a parallel monolingual treebank with semantic similarity relations, *he Sixth International Workshop on Treebanks and Linguistic Theories (TLT'07)*, Bergen, Norway,.

Nelken, Rani and Stuart M. Shieber (2006), Towards robust context-sensitive sentence alignment for monolingual corpora, *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, Trento, Italy.

Och, Franz J. and Hermann Ney (2003), A systematic comparison of various statistical alignment models, *Comput. Linguist.* **29** (1), pp. 19–51, MIT Press, Cambridge, MA, USA. http://dx.doi.org/10.1162/089120103321337421.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002), Bleu: a method for automatic evaluation of machine translation, *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 311–318.

Pedersen, Ted and Anagha Kulkarni (2006), Automatic cluster stopping with criterion functions and the gap statistic, *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 276–279. http://dx.doi.org/10.3115/1225785.1225792.

Quirk, Chris, Chris Brockett, and William Dolan (2004), Monolingual machine translation for paraphrase generation, *in* Lin, Dekang and Dekai Wu, edi-

tors, *Proceedings of EMNLP 2004*, Association for Computational Linguistics, Barcelona, Spain, pp. 142–149.

Riezler, Stefan, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu O. Mittal, and Yi Liu (2007), Statistical machine translation for query expansion in answer retrieval, *ACL*.

Russo-Lassner, Grazia, Jimmy Lin, and Philip Resnik (2006), A paraphrase-based approach to machine translation evaluation, *Technical report*, University of Maryland, College Park. http://www.umiacs.umd.edu/˜jimmylin/publications/Russo-Lassner-etal-TR-LAMP-125-2005.pdf.

Stolcke, Andreas (2002), SRILM - An Extensible Language Modeling Toolkit, *In Proc. Int. Conf. on Spoken Language Processing*, Denver, Colorado, pp. 901–904.

Zhou, Liang, Chin-Yew Lin, and Eduard Hovy (2006), Re-evaluating machine translation results with paraphrase support, *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Sydney, Australia, pp. 77–84. http://www.aclweb.org/anthology/W/W06/W06-1610.