Workshop Description

Multiword units (MWUs) include a large range of linguistic phenomena, such as phrasal verbs (e.g. "look forward"), nominal compounds (e.g. "interior designer"), named entities (e.g. "United Nations"), set phrases (e.g. "con carne") or compound adverbs (e.g. "by the way"), and they can be syntactically and/or semantically idiosyncratic in nature. MWUs are used frequently in everyday language, usually to express precisely ideas and concepts that cannot be compressed into a single word. A considerable amount of research has been devoted to this subject, both in terms of theory and practice, but despite increasing interest in idiomaticity within linguistic research, many questions still remain unanswered. The objective of this workshop is to deal with three important questions that are of great interest for real-world applications.

1) Comparison of MWU extraction methodologies

Many methodologies have been proposed in order to automatically extract or identify MWUs. However, not many efforts have been devoted to compare their results. The core differences between the methodologies are certainly the main reason why such works are so rare. For instance, it is not easy to compare language-dependent methodologies as the results depend on the efficiency of parameter tuning (i.e. semantic tagging, local specific grammars, lemmatization, part-of-speech tagging etc.). Another important problem is the fact that there is no real agreement between researchers about the definition of MWUs which would provide the basis for an objective evaluation.

2) Evaluation of the benefits of the integration of MWUs in real-world applications

It is not yet clear whether MWUs really improve NLP applications. It is common sense that Machine Translation is one application that takes great advantage of MWUs databanks. However, does the same apply to applications in Automatic Summarization, Information Retrieval (IR), Cross-language IR, Information Extraction, Text Clustering/Classification, etc.? Indeed, could the identification of MWUs introduce new constraints that are not present in original texts? Should MWUs be considered as units that should not be analyzable in terms of their components meaning? Or should they be treated as un-analyzable? Should NLP methods work both on isolated words and on aggregated MWUs? The answers are anything but clear. Here, the objective of the workshop is to point at successes and failures of the integration of MWUs in real-world applications.

3) Comparison of scalable architectures for the extraction and identification of MWUs

Real-world applications are constrained by variables like processing time and memory space. And, identifying and extracting MWUs is usually a computationally heavy process. In recent years, new algorithms and new technologies have been proposed to introduce MWU treatment in large scale applications, thus avoiding previous intractable implementations. Previous workshops on MWUs have mainly focused on the unconstrained extraction process. In this workshop, we would like to focus on the comparison of different factors that can influence the scalability of the treatment of MWUs in real-world applications, namely data structures, algorithms, parallel and distributed computing, grid computing etc.

We hope that the papers of these proceedings will reach your and our expectations.

Acknowledgments

We wish to thank all the reviewers for their remarkable work. The quality of the Workshop mainly depends on their efforts for evaluation and the final programme is the result of hard work from the Program Committee.

We wish to thank all the members of the Local Organizing Committee for their constant availability and competence.

We also want to thank all the authors for the quality of the submitted papers. Unfortunately, good papers have not been accepted due to the low acceptation rate of the workshop (38%). Thanks for them also.

Finally, we want to thank our partners, cited below, for their funding and logistic help.

Partners

Department of Computer Science of Beira Interior University

Department of Computer Science of New University of Lisbon

Faculty of Arts of Ljubljana University

Fundação Luso-Americana para o Desenvolvimento

Fundação para a Ciência e a Tecnologia

The Workshop Programme

May, 25, 2004

FIRST SESSION

Chair: Špela Vintar

9h00 - 9h45 - Kenneth W. Church – Invited Speaker

9h45 - 10h05 - T. Ogata, K. Terao and K. Umemura - Japanese Multiword Extraction using SVM and Adaptation

10h05 - 10h25 - M.C. Díaz-Galiano, M.T. Martín-Valdivia, F. Martínez-Santiago and L.A. Ureña-López - Multiword Expressions Recognition with the LVQ Algorithm

10h25 - 10h45 - R. Pereira, P.Crocker and G. Dias - A Parallel Multikey Quicksort Algorithm for Mining Multiword Units

END FIRST SESSION

10h45 - 11h00 - Coffee Break

SECOND SESSION

Chair: José Gabriel Pereira Lopes

11h00 - 11h20 - N. Kaji and S. Kurohashi - Recognition and Paraphrasing of Periphrastic and Overlapping Verb Phrases

11h20 - 11h40 - C.H.A. Koster - Transducing Text to Multiword Units

11h40 - 12h00 - J. Nivre and J. Nilsson - Multiword Units in Syntactic Parsing

12h00 - 12h20 - O. Vechtomova and M. Karamuftuoglu - Use of Noun Phrases in Interactive Search Refinement

12h20 - 12h40 - Š. Vintar - Comparative Evaluation of C-value in the Treatment of Nested Terms

12h40 - 13h00 - Discussion and Closing Session

END SECOND SESSION

Workshop Organisers

Gaël Harry Dias (Beira Interior University, Portugal)

José Gabriel Pereira Lopes (New University of Lisbon, Portugal)

Špela Vintar (University of Ljubljana, Slovenia)

Workshop Programme Committee

Timothy Baldwin (Stanford University, United States of America)

Sophia Ananiadou (University of Salford, England)

Didier Bourigault (University of Toulouse, France)

Pascale Fung (University of Science and Technology, Hong Kong)

Mikio Yamamoto (University of Tsukuba, Japan)

Dekang Lin (University of Alberta, Canada)

Aline Villavicencio (University of Cambridge, England)

Heiki Kaalep (University of Tartu, Estonia)

Joaquim da Silva (New University of Lisbon, Portugal)

Eric Gaussier (Xerox Research Centre Europe, France)

Adeline Nazarenko (University Paris XIII, France)

António Branco (Lisbon University, Portugal)

Workshop Local Organizing Committee

João Paulo Cordeiro (Beira Interior University, Portugal)

Sara C. Madeira (Beira Interior University, Portugal)

Sérgio Nunes (Beira Interior University, Portugal)

Table of Contents

Japanese Multiword Extraction using SVM and Adaptation 8
T. Ogata, K. Terao and K. Umemura

Multiword Expressions Recognition with the LVQ Algorithm 12
M.C. Díaz-Galiano, M.T. Martín-Valdivia, F. Martínez-Santiago and L.A. Ureña-López

A Parallel Multikey Quicksort Algorithm for Mining Multiword Units 17
R. Pereira, P.Crocker and G. Dias

Recognition and Paraphrasing of Periphrastic and Overlapping Verb Phrases 24
N. Kaji and S. Kurohashi

Transducing Text to Multiword Units 31
C.H.A. Koster

Multiword Units in Syntactic Parsing 39
J. Nivre and J. Nilsson

Use of Noun Phrases in Interactive Search Refinement 47
O. Vechtomova and M. Karamuftuoglu

Comparative Evaluation of C-value in the Treatment of Nested Terms 54
Š. Vintar

Author Index

Name	*Affiliation*	*Email*
C.H.A. Koster	University of Nijmegen (The Netherlands)	kees@cs.kun.nl
F. Martínez-Santiago	University of Jaén (Spain)	dofer@ujaen.es
G. Dias	Beira Interior University (Portugal)	ddg@di.ubi.pt
J. Nilsson	Växjö University (Sweden)	jens.nilsson@ msi.vxe.se
J. Nivre	Växjö University (Sweden)	joakim.nivre@msi.vxe.se
K. Church	Microsoft (United States of America)	church@microsoft.com
K. Terao	Toyohashi University of Technology (Japan)	teraken@ss.ics.tut.ac.jp
K. Umemura	Toyohashi University of Technology (Japan)	umemura@tutics.tut.ac.jp
L.A. Ureña-López	University of Jaén (Spain)	laurena@ujaen.es
M.C. Díaz-Galiano	University of Jaén (Spain)	mcdiaz@ujaen.es
M. Karamuftuoglu	Bilkent University (Turkey)	hmk@cs.bilkent.edu.tr
M. Martín-Valdivia	University of Jaén (Spain)	maite@ujaen.es
N. Kaji	University of Tokyo (Japan)	kaji@kc.t.u-tokyo.ac.jp
O. Vechtomova	University of Waterloo (Canada)	ovechtom@engmail.uwaterloo.ca
P.Crocker	Beira Interior University (Portugal)	crocker@di.ubi.pt
R. Pereira	Beira Interior University (Portugal)	rpereira@di.ubi.pt
S. Kurohashi	University of Tokyo (Japan)	kura@kc.t.u-tokyo.ac.jp
Š. Vintar	Ljubljana University (Slovenia)	spela.vintar@guest.arnes.si
T. Ogata	Toyohashi University of Technology (Japan)	tomomi@ss.ics.tut.ac.jp

Papers

Japanese Multiword Extraction Using SVM and Adaptation

Tomomi Ogata, Kenichiro Terao and Kyoji Umemura

Toyohashi University of Technology, Information & Computer Sciences
Toyohasi-shi, Aichi 441-8580, Japan
tomomi@ss.ics.tut.ac.jp, teraken@ss.ics.tut.ac.jp, umemura@tutics.tut.ac.jp

Abstract

This paper proposes the use of a Support Vector Machine (SVM) and variations in document frequency to extract Japanese technical terms from non-segmented Japanese text. Technical terms are usually multiword. This paper proposes to test whether a sequence of strings appears more than merely by chance within the same document rather than within the whole corpus. Using these statistics, rather than conventional statistics, the distribution of technical term is given a better separation from random fragments of a string. This makes it possible for SVM to extract Japanese technical terms even from non-segmented Japanese texts. This paper exploits the distribution of technical terms and random fragments of strings, and reports the level of agreement between technical terms specified by the authors and the output of the system.

Introduction

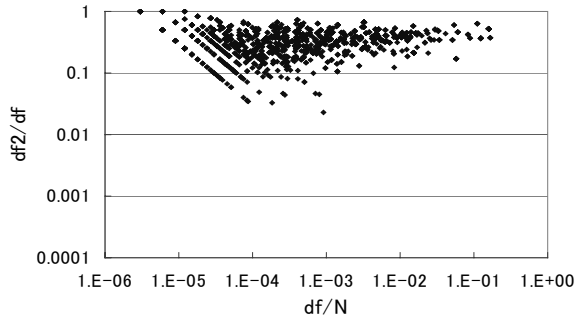
One of the possible features of multiword aggregates could be that the sequence of the words is due to more than chance alone. There are many ways to define chance. Many studies have been conducted using word frequency in the corpus, or the number of documents that contain that particular sequence of words (Lin, 1998) (Utiyama, 2000) (Nakagawa, 2002). The probability is usually measured for the whole corpus. We propose to test whether a sequence of word can be detected as due to more than chance within the same document rather than within the corpus.

The adaptation of strings is based on the probability of how frequently a string already found in the document can be observed again. This statistical technique was introduced by Church (2000), who reported that the values of adaptation for English words are usually huge compared to their probability of being detected. We have verified this property of adaptation for Japanese technical terms, which are usually multiword. Although, there are many other sophisticated statistical functions such as reprentativeness as proposed by Hisamitsu (2002), adaptation is entirely statistical.

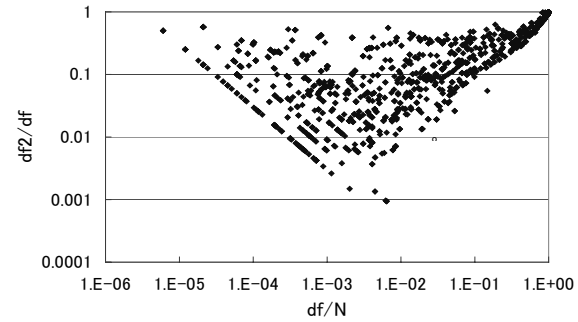
Takeda (2001) has used such statistics to extract technical terms from Chinese and Japanese texts, and has shown that the extracted terms improve search effectiveness. Takeda's method uses the Viterbi search to segment a Japanese text and employs predefined values for both to statistical segmentation and to judge whether or not each segmented string is a technical term. Since the predefined values apparently depend on the corpus, Takeda's method is hard to replicate.

Our method uses the adaptation of strings as an input feature of SVM (Vapnik, 1995) to judge whether or not given string is a multiword technical term. In addition to the value for adaptation of given string, we have also decided to use adaptation for a string with one additional character. This decision comes from the fact that the value of adaptation is stable within multiwords, and that the value declines with the additional character. Takeda (2001) first reported this fact. We have also used document frequency when considering the multiword distribution.

The input of our method consists of a sample of technical terms and a sample of string fragments, resulting in two classes of samples, SVM learn hyper-plane in feature space between technical terms and fragments. After learning this hyper-plane, the SVM will output the strings that belong to the category of technical terms. In this method, there is no predefined threshold value for adaptation. The SVM will learn the threshold value from the samples.



Technical Terms



String Fragments

Figure 1 : df/N vs. df_2/df plots show significantly different shapes between terms and fragments.

The experimental results show that our new method achieves better recall compared with Takeda's method, and that it is essential to use adaptation as an input feature of SVM. In contrast to Takeda's method, ours does not contain the threshold value for statistics, thus making it easy to replicate our results. We conclude that SVM and adaptation constitute a new baseline system to extract multiword Japanese technical terms.

Adaptation and its Related Estimators

Let N be the number of documents in the corpus we have to deal with. Let P(tf(D,s)≥1) be the probability of detecting a certain *s* in randomly selected documents. The expectation of P(tf(D,s)≥1) can be estimated by df(s)/N, where df(s) is the number of documents which contain the string more than once. It should be noted that we have chosen the document rather than the string as the random variable, even though our parameter is the string. It should also be noted that *s* may be not only a word but also any fragment of a string.

The adaptation of a string *s* is defined as the expectation of P(tf(D,s)≥2|tf(D,s)≥1). Again, It should be noted that we have chosen the document rather than the string as the random variable, even though our parameter is the string. This probability can be estimated by $df_2(s)/df(s)$, where $df_2(s)$ is the number of documents which contain the string *s* two or more times.

$$
\begin{aligned}
&E\{P(tf(D,s) \geq 2) \mid tf(D,s) \geq 1\}\} \\
&= E\{P(tf(D,s) \geq 2 \ \& \ tf(D,s) \geq 1) / P(tf(D,s) \geq 1)\} \\
&= E\{P(tf(D,s) \geq 2) / P(tf(D,s) \geq 1)\} \\
&\cong (df_2(s)/N) / (df(s)/N) \\
&= df_2(s)/df(s)
\end{aligned}
$$

Church pointed out that almost all English words show large adaptation. Takeda also verified this for Chinese and Japanese texts, and also reported that the adaptation of Japanese technical terms is large. We can imagine that even a rare multiword may have large adaptation. For example, although "MEMURA workshop" is a very rare multiword, if there are certain documents that discuss the workshop, we may expect "MEMURA workshop" to appear several times in those documents. This makes the adaptation of "MEMURA workshop" much larger than its df/N. Takeda (2001) also reported that the adaptation of string fragments of Japanese is very different from that of technical terms. We have verified that finding a shown in figure 1. We are motivated to use SVM to separate technical terms from string fragments because their region shows almost the same separation as that seen in figure 1.

Adaptation of Multiword Substrings

Takada (2001) also includes figure 2, which shows the adaptation of a substring of the multiword "President Fujimori", and pointed out that the first dropping point is on the boundary of this multiword. This implies that all substrings of the multiword may have an large adaptation. Takeda (2001) also noted the fact that the adaptation does not drop between "Fujimori" and "President", where df/N will have a dropping point. In the figure, the third word from the right is the right term. The second word has an additional functional word, and the first one has an additional comma. Takeda (2001) claims that the adaptation has the information to distinguish the word boundary from the string fragment.

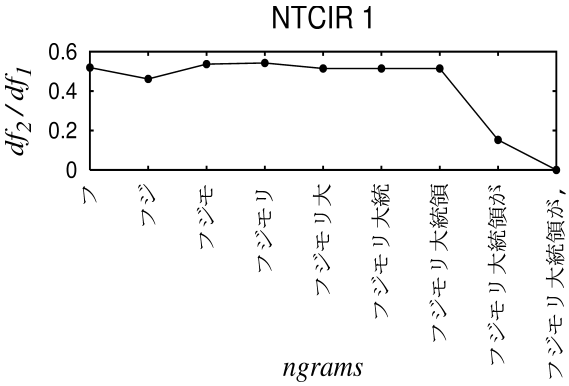
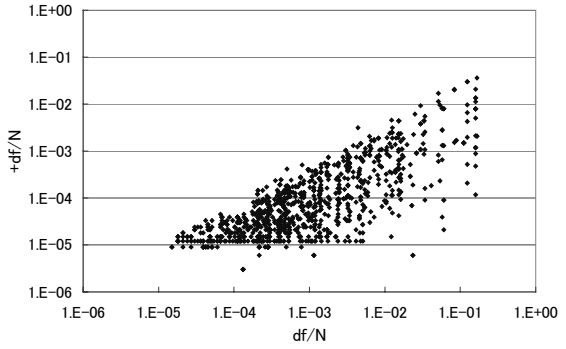
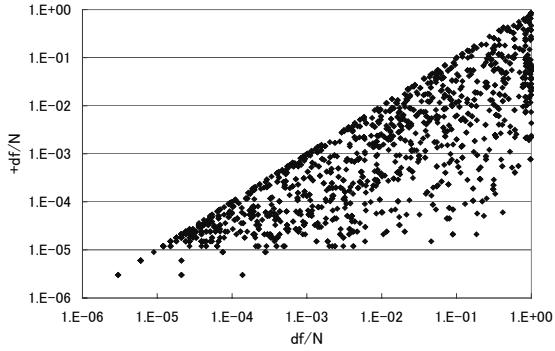


Figure 2 : The adaptation ratio df_2/df_1 drops rapidly on the first character after the word boundary. (Takeda, 2001)



Technical Term (df/N , $^{+}$df/N)



String Fragment (df/N , $^{+}$df/N)

Figure 3 : The points (df/N , $^{+}$df/N) of technical terms show the drop on the boundary, but the region is highly overlapped with that of string fragments. The same relation exists for (df/N , df^{+}/N)

Boundary Information by df/N

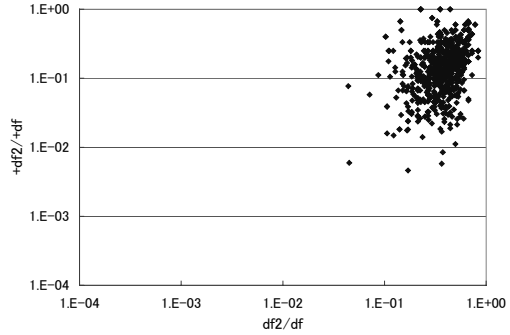
Figure 2 suggests that all substrings of a multiword may have an large adaptation. This means that SVM cannot distinguish the substrings from the original multiword. To capture the boundary of a multiword, we first compute the $^{+}$df(s) and df^{+}(s), where the $^{+}$df(s) is the number of documents that contain the preceding one character and *s*

together, the $df^+(s)$ is the number of documents that contain *s* and the following character together.

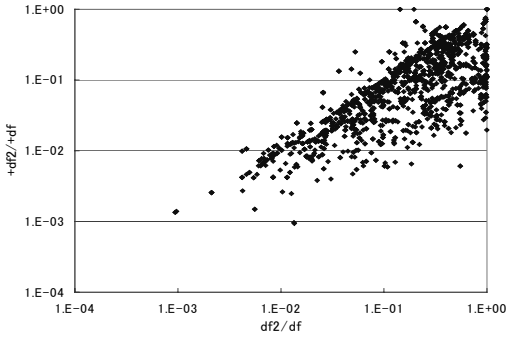
Figure 3 shows that the technical term show drops on the boundary and that since there are very few points of $df(s)/N \cong {}^+df(s)/N$, $df(s)/N \cong df^+(s)/N$ for technical terms and many for string fragments, the region is greatly overlapped. This suggests that this feature may not be suitable for SVM.

Boundary Information by $df_2(s)/df(s)$

Then we compute the ${}^+df_2(s)/{}^+df(s)$ and $df_2^+(s)/df^+(s)$ where the ${}^+df_2(s)$ is the number of documents that contain the preceding one character and *s* together two or more times. The $df_2^+(s)$ is the number of documents that contain *s* and the following character together two or more times.



Technical Term (df_2/df, ${}^+df_2/{}^+df$)



String Fragments (df_2/df, ${}^+df_2/{}^+df$)

Figure 4 : The points (df_2/df , ${}^+df_2/{}^+df$) of technical terms shows the drop on the boundary, with the region less overlapped with string fragment than Figure 3. The same relation is observed for the points (df_2/df , df_2^+/df^+)

Figure 4 also shows that there are very few points of $df_2(s)/df(s) \cong {}^+df_2(s)/+df(s)$, $df_2(s)/df(s) \cong df_2^+(s)/df^+(s)$ for technical terms, and many for string fragments. We can also observe that the overlapping region is narrower than that in figure 3. This suggest that ${}^+df_2/{}^+df$ and df_2^+/df_+ may be a better feature for SVM than ${}^+df/N$ and df+/N

Experiment

We have conducted our experiment using the NTCIR-1 test collection (Kando, 2000), which consists of around 330,000 samples of technical abstracts. This collection contains lists of technical terms as keywords provided by the corresponding authors, allowing us to compare the output of our method with those terms. In this test collection, about 83% of these terms are multiword technical terms. We have used 1000 abstracts for training and another 500 for testing.

The training set of SVM contains around 1000 randomly selected technical terms for the positive set and 3000 fragments of randomly selected terms for the negative set. We had to select the terms because there were too many negative examples to learn, and such a huge number of samples makes learning difficult. Since this is a large number of terms for the simple implementation of SVM, we have chosen SVM^{light} (Joachims, 1999). This implementation is known to be capable of dealing with many thousands of support vectors efficiently. SVM^{light} is freely available for research purposes.

The test set of SVM consists of another 1808 technical terms and another 666382 fragments. We have chosen technical terms from the author's keywords in the abstracts, and have chosen fragments that appear in the corpus three or more times. We did not select multiwords from the technical terms for two reasons. The first is the difficulty of deciding whether or not a technical term in Japanese is a multiword. Had we chosen multiwords from the technical terms, the experimental results would have been influenced by subjective human judgment. The second is that we can treat both kinds of technical terms in the same manner in almost all applications.

We are interested in what kind of feature set is suitable for this task. The first baseline is called DF-ONLY. The input features of SVM are df/N, df^+/N and ${}^+df/N$. Since many existing methods are based on the df/N, we have established it as the counter part of our system. The second baseline is DF2/DF-ONLY. This baseline is intended to compare the power of df/N and df_2/df. Its input features are df_2/df, df_2^+/df^+, and ${}^+df_2/{}^+df$. The third system is one whose features are df/N and df_2/df. Since this only contains information on the base form, we call it BASE-ONLY. This may not work well, however, since all the substrings of technical words are also classified as a positive set. Our system is called PROPOSED, and its input features include all the above.

System	Output#	Correct#	Recall	Precision	F-value
DF-ONLY	0	0	0.000	0.000	0.000
DF2-ONLY	18526	281	0.155	0.015	0.028
BASE-ONLY	1364	61	0.045	0.034	0.038
PROPOSED	59940	1561	0.863	0.026	0.051

Table 1 : All proposed features contributing to the performance of the system

As we predicted, DF2-ONLY can output some correct terms, but DF-ONLY can do nothing. All strings are regarded as non-technical terms. PROPOSED is better than DF2-ONLY, although DF-ONLY is not working at all. This is due to the presence of a rather clear separation line, as seen in figure 1.

Even though SVM is known for its robust statistics, the performance of the system depends on the size of the

learning set. To verify that we have accumulated a sufficient number of samples, we examined the performance of our system, adjusting the size of the learning set when needed. Table 2 shows that the number of learning sets appears to be sufficient.

Learning #	Output#	Correct#	Recall	Precision	F-value
100	93071	1557	0.861	0.0167	0.0328
300	94111	1593	0.881	0.0169	0.0332
500	69768	1549	0.857	0.0222	0.0433
1000	59940	1561	0.863	0.0260	0.0505

Table 2 : The size of the learning set is sufficient

Discussion

The results of our system are still not satisfactory in terms of precision. Many of the outputs are parts of technical terms. This suggests that the system needs further features that will detect about the boundary. We are estimating that the overlapped region in figure 3 is the cause of the problem

Both Takeda (2001) and Fung (1998) attempted to detect technical terms for application to information retrieval. Compared to Fung's, method, Takeda's made very few assumptions, and our method makes even fewer assumptions than Takeda's. Since in contrast to his, ours does not contain the threshold value for statistics, thus facilitating the replication of results, the precision of our method is worse, but the recall is better than that in Takeda's method. It is difficult to say which is the better method since they behave so very differently. One of the key advantages of our method lies in the fact that the system does not need to be modified even when a new corpus becomes available. We may thus conclude that SVM and adaptation constitute a new baseline system to extract multiword keywords from Japanese texts.

System	Output#	Correct#	Recall	Precision	F-value
PROPOSED	59940	1561	0.863	0.026	0.051
TAKEDA	2887	199	0.178	0.069	0.100

Table3 : Comparison with Takeda's method

Utiyama (2000) used the same corpus to recognize the terms, and employed authors' technical terms as training data. His system arrives at the statistical distribution of terms and distinguishes terms from functional words, and differs further in using a segmented corpus. Apparently, selecting terms from a non-segmented text is more difficult than selecting them from a segmented word. Utiyama also reported that the F-value was more than 0.5, which is much higher than the one in our system. However, many of our errors arose from badly segmented words, and from the fact that a segmented corpus may not always be available.

From another point of view, the task of detecting technical terms without using a dictionary has proven to be a difficult one, but adaptation has been able to provide informative statistics to make it easier.

Conclusion

This paper demonstrates that SVM and adaptation together have the ability to select technical term from Japanese documents without resort to any kind of dictionary. By choosing a baseline that does not use adaptation at all, adaptation proved to be an essential source of information. Unlike previous methods, this method contains no heuristics and is easy to replicate.

Acknowledgement

The research was based on the IPA Unexplored Software Project for the 2000-2001 fiscal year. This research was supported by Sumitomo Electric Industries, Ltd and Grant-in-And for Scientific Research (C), No. 15500090, the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

- Dekang Lin (1998). Extracting Collocations from Text Corpora. In First Workshop on Computational Terminology.
- Hiroshi Nakagawa & Tatunori Mori. (2002). A Simple but Powerful Automatic Term Extraction Method, 2nd Workshop on Computational Terminology (pp. 29—35)
- Kenneth W. Church. (2000). Empirical Estimates of Adaptation: The Chance of Two Noriegas is closer to p/2 than p^2, In Coling-2000 (pp. 180—186).
- Kyo Kageura. (1996). Methods of Automatic Term Recognition, Terminology, Vol. 3, No. 2, (pp. 259—289).
- Masao Utiyama, Masaki Murata & Hitoshi Isahara. (2000). Using Author Keywords for Automatic Term Recognition, Terminology Vol. 6, No. 2 (pp.313—326)
- Noriko Kando. (2000). NTCIR Project. http://research.nii.ac.jp/ntcir/.
- Pascale Fung. (1998). Extracting Key Terms from Chinese and Japanese texts. In International Journal on Computer Processing of Oriental Languages, Special Issue on Information Retrieval on Oriental Languages, Vol.12, No. 1 (pp. 99—121).
- Thorsten Joachims. (1999). Making Large-scale SVM Learning Practical. Advances in Kernel Methods – Support Vector Learning. In B. Schoeklopf, C. Burges and A. Smola (ed.), MIT Press.
- Toru Hisamitsu & Yoshiki Niwa. (2002). A Measure of Term Representativeness Based on the Number of Co-occurring Salient Words, COLING.
- Vladimir N. Vapnik. (1995). The Nature of Statistical Learning Theory. Springer.
- Yoshiyuki Takeda & Kyoji Umemura. (2001). Selecting Indexing Strings Using Adaptation. In Proceedings of the 25th Annual International ACM SIGIR Conference. (PP. 427—428).

Multiword Expressions Recognition with the LVQ Algorithm

M.C. Díaz-Galiano, M.T. Martín-Valdivia, F. Martínez-Santiago, L.A. Ureña-López
Departamento de Informática. University of Jaén.
E-23071. Spain
{mcdiaz, maite, dofer, laurena}@ujaen.es

Abstract

This paper proposes a new neural method based on the supervised Kohonen model for multiword expressions recognition. We use the Learning Vector Quantization algorithm to integrate several statistical estimators to solve this task. Lists of multiword expressions and non-multiword expressions have been generated using the WordNet lexical database to train and test the neural network. Then the neural net has been applied to recognise multiword expressions in a monolingual corpus to prove the effectives in an information retrieval system. The results show that the proposed method is an effective alternative to multiword expressions recognition task.

Introduction

In recent years, there has been growing interest in the Multiword Expressions (MWEs) recognition problem. MWEs are formed by various terms that usually express ideas and concepts that cannot be compressed into a single word. MWEs recognition is very important in many NLP tasks (e.g. machine translation, question-answering, summarisation, etc.). Most real-world applications tend to ignore MWEs or address them simply by listing. However, it is clear that successful applications will need to be able to identify and treat them appropriately.

Methods for automated MWEs recognition have traditionally been statistical (Hull, 1996), (Ballesteros, 1998), and based on the co-occurrence of each particular pair of words in the corpus. Other works (Adriani, 1999) obtain the degree of similarity between terms using the co-occurrence factor, and the standard *tf·idf* term weighting formula. Recently, hybrid approaches incorporating linguistic information have been developed: Diana Maynard and Sophia Ananiadou (Maynard, 2000) make use of different types of contextual information: syntactic, semantic, terminological and statistical. However, different types of information must be managed by integrating them in a given way. The most straightforward is by using a linear function, although this may not be the best way to tackle the problem.

We propose a well-known supervised neural network: Kohonen's Learning Vector Quantization (LVQ) widely used for classification tasks (Kohonen, 1995). The LVQ algorithm will be used to integrate several statistical estimators in order to recognise MWEs.

The rest of the paper is organized as follows. Firstly, we present an introduction to the state of the art, briefly showing some of the currently available methods used to MWEs recognition. These methods include different estimators that will be lately used in our approach. Then we describe the neural network architecture used and the LVQ algorithm. Next section shows the experiments carried out and the results obtained. Finally, we present some conclusions and lines of future work.

Statistical Estimators

Most works what attempt to solve the MWEs recognition problem use estimators to classify terms group. We have integrated several of these estimators, which obtain good results separately, in a neuronal network, trying select a set of heterogeneous and representative estimators.

We have used the following estimators to train and test the neural net:

1. *Pearson's* χ^2. A variant of the χ^2 statistic (Hull, 1996).
2. The *mutual information ratio*, or association ratio, μ (Johansson, 1996).

$$\mu = \log_2\left(\frac{P_{xy}}{P_x \cdot P_y}\right)$$

where P_i is the ocurrence probability of term i in the corpus. This probability is calculated as:

$$P_i = \frac{F_i}{T}$$

where:
F_i = frecuency occurrence of term i.
T = total number of term in the corpus.
Therefore, the first formula can write as:

$$\mu = \log_2\left(\frac{T \cdot F_{xy}}{F_x \cdot F_y}\right)$$

3. Measure the importance of co-occurrence of the elements in a set by the *em* metric (Ballesteros, 1998).

$$em_{xy} = \max\left(\frac{F_{xy} - En(x,y)}{F_x + F_y}, 0\right)$$

where:

$$En(x,y) = \frac{F_x \cdot F_y}{T}$$

where T is the total number of terms in the corpus.

4. *Dice similarity coefficient* obtain the degree of similarity or association-relation between terms using a term association measure and the tf.idf weighting formula (Adriani, 1999).

$$Dice_{xy} = 2\frac{\sum_{i=1}(w'_{xi} \cdot w'_{yi})}{\sum_{i=1} w_{xi}^2 + \sum_{i=1} w_{yi}^2}$$

where:
w_{xi} = the weight of term x in the document i.
w_{yi} = the weight of term y in document i.
$w'_{xi} = w_{xi}$ if term y also occurs in document i, or 0 otherwise.
$w'_{yi} = w_{yi}$ if term x also occurs in document i, or 0 otherwise.
n = the number of documents in the collection.

5. *Dice similarity coefficient*. A variant of the *Dice* estimator (Martinez, 2002).

$$xy = 2\frac{\sum_{i=1}(w'_{xi} \cdot w'_{yi})}{\min\left(\sum_{i=1} w_{xi}^2, \sum_{i=1} w_{yi}^2\right)}$$

Neural Network Architecture

A Neural Network (NN) is a statistical information model that uses learning to adjust the model. NN has been successfully applied in many NLP tasks. In this paper, we propose the use of a competitive neural learning model based on the supervised Kohonen model (Kohonen, 1995) to accomplish the MWEs recognition task: the LVQ algorithm.

MWEs detection is a categorization problem in which only two categories have to be managed: MWE and non-MWE. In our experiment only multiwords with two relevant terms have been used. Consequently, classifying a pair of terms turns into a two step process: firstly, obtain the values yielded by the different estimators; secondly, use those values as inputs for the neural network, and obtain the class to which the pair of terms belongs. More precisely, we have used 5 estimators: *Pearson's* χ^2 (Hull, 1996), the *em* metric (Ballesteros, 1998), the *Dice similarity coefficient* (Adriani, 1999), the *mutual information ratio*, μ (Johansson, 1996) and the *Simpson* coefficient. Figure 1 shows the network architecture.

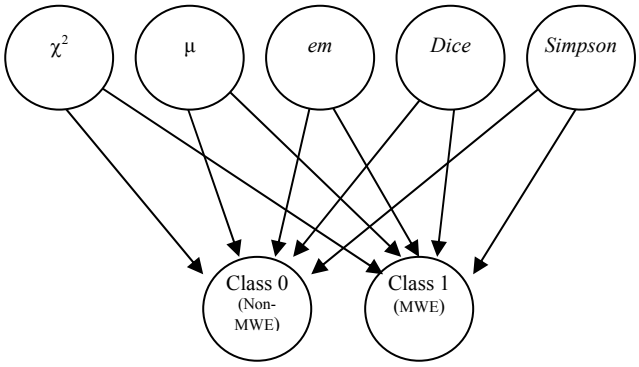


Figure 1: Neural network architecture for MWEs recognition task

In order to train and test our neural network approach, a set of patterns composed of input-output pairs had to be generated, every pattern corresponding to a pair of terms. Thus, to carry out the experiments, we have generated two lists of samples: One list contains vectors with values for the considered estimators for pairs of terms that are MWEs and second list contains vectors with values for the considered estimators for pairs of terms that are non-MWEs. Each vector is labelled with the class to which it belongs (class 1 for the vectors that belong to the MWEs class and class 0 for the vectors that belong to the non-MWEs class).

We have used the CLEF[1] 2001 collection data in order to generate both samples lists. The values for MWEs were obtained by applying the considered estimators to pairs of terms present in the corpus that are MWEs, and labelling the output with the class 1. We consider that a pair of terms is a MWEs if it appears in WordNet (Miller, 1995). WordNet is a lexical database where MWEs can be found. However, not all the pairs of terms said to be MWEs really were. For this reason, each MWE returned by WordNet was checked against in the machine readable dictionary Encarta[2] to remove pairs of terms which were not real MWEs, even though they appeared together very frequently.

A Non-MWE list was also taken from the corpus used in CLEF 2000. Pairs of terms were taken from this corpus and then searched in WordNet, checking that they did not appear in it. If they did not appear, they were once more

[1] The Cross Language Evaluation Forum (CLEF) is an annual activity of European ambit, held since 2000 and coordinated by DELOS Network of Excellence for Digital Libraries conferences, in collaboration with the NIST and the TREC. CLEF aims to promote research and development in CLIR. For more information, see: http://www.clef-campaign.org.

[2] Encarta is a machine readable dictionary available at http://www.encarta.com.

checked against the Encarta dictionary to assure they were not MWEs. The output values for non- MWEs were labelled with the 0 class.

The LVQ Algorithm

To train and test the neural network we have used the supervised Kohonen model: the LVQ algorithm (Kohonen, 1995). This learning algorithm is a classification method based on neural competitive learning, which permits the definition of a group of categories in the space of input data by a reinforced learning. LVQ uses supervised learning to define class regions in the input data space. To this end, a subset of similarly labelled codebook vectors is placed into each class region.

Given a sequence of input data, an initial group of reference vectors w_k (codebook) is selected. In each iteration, a input vector x_i is selected and the vectors W are updated, so that they fit x_i in a better way. The LVQ algorithm works as follows:

For each class, k, a weight vector w_k is associated. In each repetition, the algorithm selects an input vector, x_i , and compares it with every weight vector, w_k, using the euclidean distance $||x_i-w_k||$, so that the winner will be the weight vector w_c nearest to x_i, being c its index:

$$\left\| x_i - w_c \right\| = \min_k \left\{ \left\| x_i - w_k \right\| \right\}$$

i.e.,

$$c = \arg\min_k \left\| x_i - w_k \right\|$$

The classes compete between themselves in order to find the most similar to the input vector, so that the winner is the one with shorter Euclidean distance with regard to the input vector. Only the winner class will modify its weights using a reinforced learning algorithm, either positive or negative, depending on the classification being correct or not. Thus, if the winner class and the input vector have the same class (the classification has been correct), it will increase the weights, coming slightly closer to the input vector. On the contrary, if the winner class is different from the input vector class (the classification has not been correct), it will decrease the weights, moving slightly further from the input vector.

Let $x_i(t)$ be an input vector at time t, and $w_k(t)$ represents the weight vector for the class k at time t. The following equation defines the basic learning process for the LVQ algorithm.

$$w_c(t+1) = w_c(t) + s \cdot \alpha(t) \cdot \left[x_i(t) - w_c(t) \right]$$

where $s = 0$, if $k \neq c$; $s = 1$, if $x_i(t)$ and $w_c(t)$ belong to the same class; and $s = -1$, if they do not, and where $\alpha(t)$ is the learning rate, being $0<\alpha(t)<1$, a monotonically decreasing function of time. In our experiments we have used $\alpha(t) = 0.3$.

Once the training phase has finished, the production phase starts. Again, each testing vector is presented to the network input. The original LVQ algorithm must find the winner class calculating the Euclidean distances between the codebook vectors and input vectors. The winner class will be the codebook vectors with the shortest distance with regard the input vector.

In order to improve the network precision, we have used a modified version of the LVQ algorithm during the evaluation phase. An input vector is presented to the neural network but the output network is the distance to the codebook vector belonging to the MWEs class (class labelled with 1). This value represents a confidence score assigned by the neural network to the pair of terms considered is a MWE. Thus, during the evaluation the non-MWE class is no considered. The network output is normalized and finally is inverted by subtracting 1. Thus, value near to 1 indicates a high confidence in the input vector represents a MWE.

Evaluation and Results

As we have commented previously, the experiments have been carried out for the English CLEF 2001 collection data. The collection is composed by 113,005 news from the *Los Angeles Time* newspaper edition 1994, 50 queries and their relevance assessments.

The corpus has been pre-processed as usual in information retrieval systems (Frakes and Baeza-Yates, 1992), using stopword lists and stemming algorithms available via the Web[3]. Stopword lists have been increased with terms such as “retrieval”, “documents”, “relevant”, etc.

Next step consists of generating the two lists of samples from corpus. Once both MWEs and non-MWEs lists had been created, the estimators were applied to them, obtaining the file with the patterns to be used with the supervised network. This file was split to use 50% of the patterns to train the neural network and the remaining 50% to test it. A total of 1,000 training vectors and 1,000 test vectors were generated.

To test the neural network we have modified the LVQ algorithm. The original LVQ must find the winner class by calculating the Euclidean distances between the codebook vectors and input vectors. The winner class will be the codebook vectors with the shortest distance with regard to the input vector.

The experiments were carried out using the implementation described in LVQ_PAK documentation (Kohonen, 1991) with default parameters. Thus, every experiment used two codebook vectors (one per class) and the learning rate started at 0.3.

The experiments were carried out with the original LVQ and with the modified proposed version for 4 confidence values (0.95, 0.90, 0.80, 0.70). The original LVQ network recognises correctly 684 MWE over 1000 test vectors. Therefore, the original LVQ obtain a 68.40% of precision. The results of the LVQ modified are more hopeful. Table 1 shows the precision obtained when we consider only the test vectors for which the network output overcomes the confidence values. With a confidence value of 0.95 the precision obtained is a 100%. In proportion to the threshold is smaller, the precision also decrease. For 0.90 of threshold the precision is 94.84%, only 74 good MWEs are recognised. With confidences values of 0.80 and 0.70 the precisions are 88.59% and 81.10% respectively. The precision obtained increase when the confidence value increase.

[3] http://www.unine.ch/info/clef

	Patterns considered	Successfully MWE detected
LVQ-0.95	40	40
LVQ-0.90	78	74
LVQ-0.80	184	163
LVQ-0.70	291	236

Table 1: Precision obtained with several confidence values

The results obtained are very good separately, which demonstrates that LVQ network works good in the MWEs recognition task with the sample data available. In order to prove the effectives in an information retrieval system, we have indexed the CLEF 2001 English collection.
Once the collection has been pre-processed and the MWEs have been recognised, we indexed the corpus with the Zprise information retrieval system[4], using the OKAPI probabilistic model (Robertson, Walker and Beaulieu, 2000). We have generated 6 different indexes

- Index without MWEs (baseline). We use the word as indexation unit.
- Index with MWEs recognised by the original LVQ network (binary output). We use words and MWEs recognised by the neural network as indexation unit.
- Index with MWEs recognised by the modified LVQ evaluation algorithm with a confidence value of 0.95. The indexation units are words and MWEs recognised by the neural network using the modified LVQ algorithm. In this case, MWEs are considered only those that overcome the confidence value of 0.95.
- Index with MWEs recognised by the modified LVQ evaluation algorithm with a confidence value of 0.90. The indexation units are words and MWEs recognised by the neural network using the modified LVQ algorithm. In this case, MWEs are considered only those that overcome the confidence value of 0.90.
- Index with MWEs recognised by the modified LVQ evaluation algorithm with a confidence value of 0.80. The indexation units are words and MWEs recognised by the neural network using the modified LVQ algorithm. In this case, MWEs are considered only those that overcome the confidence value of 0.80.
- Index with MWEs recognised by the modified LVQ evaluation algorithm with a confidence value of 0.70. The indexation units are words and MWEs recognised by the neural network using the modified LVQ algorithm. In this case, MWEs are considered only those that overcome the confidence value of 0.70.

The experiments have been carried out by using about 105.000 news published in Los Angeles Times for 1994. In order to evaluate such experiments, we have used 50 queries and their relevance assessments from CLEF 2001 workshop. Only Title and Description sections have been taking into account. Finally, we have built six indices according to the patterns considered (without MWEs, LVQ, LVQ-0.95, LVQ-0.90, LVQ-0.80 and LVQ-070). Table 2 shows the precision obtained with the 6 indexes considered.

	Precision
Without MWEs	0.458
LVQ	0.424
LVQ-0.95	0.509
LVQ-0.90	0.471
LVQ-0.80	0.461
LVQ-0.70	0.427

Table 2. Precision obtained with MWEs recognition in an information retrieval system

The obtained results aim MWEs are useful in the information retrieval task only when precision is very high. The reason: whether erroneous MWEs are labelled then the precision achieved by the information retrieval system precision falls off dramatically. Thus, high precision is preferable even some real multiword expressions are not taking into account by the information retrieval system.

Conclusions and Future Works

We have proposed a new neural approach to MWEs recognition. The neural network uses the Kohonen LVQ algorithm. To train and test the network we have generated two lists of sample of MWEs and non-MWEs. When we evaluate the neural networks separately, the results obtained are very promising. But it does not happen the same way when the network is applied to MWEs recognition in an information retrieval system since the performance is seriously damaged by erroneous MWEs.
We could apply the proposed MWEs recognition method to improve precision in other natural language processing tasks such as summarization system, machine translation or question-answering. These tasks need a high precision in the MWEs recognition.

Acknowledgements

This work has been supported by Spanish Government (MCYT) with grant TIC2003-07158-C04-04.

References

(Adriani, 1999) M. Adriani, C.J. van Rijsbergen. Term Similarity Based Query Expansion for Cross Language Information Retrieval. *Proceedings of Research and Advanced Technology for Digital Librarie*s, 311-322, 1999.

(Ballesteros, 1998) L. Ballesteros, W.B. Croft. Resolving ambiguity for cross-language retrieval. *Proceedings of the 21st Annual International ACM SIGIR Conference*, 64-71, 1998.

(Frakes and Baeza-Yates, 1992) W.B. Frakes, R. Baeza-Yates. Information retrieval: Data, structures and algorithms. Pretice Hall, 1992.

[4] Zprise is an information retrieval system developed by Darrin Dimmick (NIST). Available on demand at http://www.itl.nist.gov/iaui/894.02/works/papers/zp2/zp2.html

(Hull, 1996) D.A. Hull, G. Grefenstette. Experiments in Multilingual Information Retrieval. *Proceedings of the 19th Annual International ACM SIGIR Conference*, 1996

(Johansson, 1996). C. Johansson. Good Bigrams. *Proceedings COLING-96*. 592-597, 1996

(Kohonen, 1995) T. Kohonen, Self-Organization and Associative Memory. 2nd Ed. Springer-Verlag, Berlin, 1995.

(Kohonen, 1996) T. Kohonen, J. Hynninen,J. Kangas, J. Laaksonen, K. Torkkola. LVQ_PAK: The Learning Vector Quantization program package. *Technical Report A30*, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, 1996.

(Maynard, 2000) D. Maynard, S. Ananiadou. TRUCKS: a model for automatic term recognition, *Journal of Natural Language Processing*, 2000.

(Miller, 1995) G. Miller. WORDNET: A lexical database for English. *Communications of the ACM*, 38 (11), 1995.

(Martínez, 2002) F. Martínez, M.T. Martín, V.M. Rivas, M.C. Díaz, L.A. Ureña. Using Neural Networks for Multiword Recognition in IR. *Proceedings of the 7th Internacional ISKO Conference*, 2002.

(Robertson, Walker and Beaulieu, 2000) Robertson, S.E., S. Walker y M. Beaulieu. 2000. Experimentation as a way of life: okapi at TREC. *Information Processing and Management*. Vol, 1, pp. 95-108

A Parallel Multikey Quicksort Algorithm for Mining Multiword Units

Rui Pereira, Paul Crocker, Gaël Dias

Beira Interior University, Computer Science Department
Rua Marquês d'Ávila e Bolama, 6201-001, Covilhã, Portugal
{rpereira, crocker, ddg}@di.ubi.pt

Abstract

This paper describes a parallel algorithm to compute positional ngram statistics based on masks and suffix arrays. Positional ngrams are ordered sequences of words that represent continuous or discontinuous substrings of a corpus. In particular, the positional ngram model has shown successful results for the extraction of discontinuous collocations from large corpora. However, its computation is heavy. For instance, 4.299.742 positional ngrams (n=1..7) can be generated from a 100.000-word size corpus in a seven-word size window context. In comparison, only 700.000 ngrams would be computed for the classical ngram model. It is clear that huge efforts need to be made to process positional ngram statistics in reasonable time and space. For that purpose, we propose a parallel algorithm based on the concept of Parallel Sorting by Regular Sampling (PSRS) described in (Shi and Schaeffer, 1992).

Introduction

In the context of word associations, multiword units (sequences of words that co-occur more often than expected by chance) are frequently used in everyday language, usually to precisely express ideas and concepts that cannot be compressed into a single word. For instance, [Bill of Rights], [swimming pool], [as well as], [in order to], [to comply with] or [to put forward] are multiword units. As a consequence, their identification is a crucial issue for applications that require a certain degree of semantic processing (e.g. machine translation, information extraction, information retrieval or summarization). In order to identify and extract multiword units, (Dias, 2002) has proposed a statistically-based architecture called SENTA (Software for the Extraction of N-ary Textual Associations) that retrieves, from text corpora, relevant contiguous and non-contiguous sequences of words.

However, the computation of SENTA is heavy. As it is based on positional ngrams (ordered sequences of words that represent continuous or discontinuous substrings of a corpus computed in a (*2.F+1*)-word size window context), the number of generated substrings rapidly explodes and reaches astronomic figures. (Dias, 2002) shows that Δ positional ngrams can be computed in an *N*-size corpus for a (*2.F+1*)-size window context (See Equation 1).

$$\Delta = (N - 2.F) \times \left(1 + F + \sum_{k=3}^{2.F+1} \sum_{i=1}^{F} \sum_{j=1}^{F} C_{j-1}^{i-1} C_{j}^{k-i-1} \right)$$

Equation 1: Number of positional ngrams

So, for instance, 4.299.742 positional ngrams would be generated from a 100.000-word size corpus in a seven-word size window context. It is clear that huge efforts need to be made to process positional ngram statistics in reasonable time to tackle real world applications that deal with Gigabytes of data. For that purpose, (Gil and Dias, 2003) have proposed an implementation that computes positional ngrams statistics in *O(h(F) N log N)*[1] time complexity based on the *Virtual Corpus* approach introduced by (Kit and Wilks., 1998). In particular, they apply a suffix-array-like method, coupled to the multikey quicksort algorithm (Bentley and Sedgewick, 1997) to compute positional ngram frequencies. Although their C++ implementation, realized over the CETEMPúblico[2] corpus, has shown satisfactory results by taking 8.34 minutes to compute the positional ngram frequencies for a 1.092.723[3]-word corpus on an Intel Pentium III 900 MHz Personal Computer for a seven-word size window context, improvements still need to be made.

So, in this paper, we propose a parallel multikey quicksort algorithm that allows faster computation of positional ngrams frequencies taking into account the processing power of different central units spread over a network. In particular, we propose a parallel algorithm based on Parallel Sorting by Regular Sampling (PSRS) as described in (Shi and Schaeffer, 1992) that apply their method to randomly generated 32 bit integers and use the classical quicksort (Hoare, 1962) as the sequential sorting algorithm. For a variety of shared and distributed memory architectures, their results display better than half linear speedups. In the following sections, we will present our PSRS algorithm that sorts positional ngrams using the multikey quicksort as the sorting algorithm.

This article is divided into four sections: (1) we explain the basic principles of positional ngrams and the mask representation to build the *Virtual Corpus*; (2) we present the suffix-array-based data structure that allows counting occurrences of positional ngrams; (3) we explain our PSRS algorithm; (4) we present some results.

Positional Ngrams

Principles

The original idea of the positional ngram model comes from the lexicographic evidence that most lexical relations associate words separated by at most five other words (Sinclair, 1974). As a consequence, lexical relations such as collocations can be continuous or discontinuous

[1] N is the size of the corpus and F is the window size.

[2] The CETEMPúblico is a 180 million-word corpus of Portuguese. It can be obtained at http://www.ldc.upenn.edu/.

[3] This represents 46.986.831 positional ngrams.

sequences of words in a context of at most eleven words (i.e. *5* words to the left of a pivot word, *5* words to the right of the same pivot word and the pivot word itself). In general terms, a collocation can be defined as a specific[4] continuous or discontinuous sequence of words in a (*2.F+1*)-word size window context (i.e. *F* words to the left of a pivot word, *F* words to the right of the same pivot word and the pivot word itself). This situation is illustrated in Figure 1 for the collocation *Ngram Statistics* that fits in the window context.

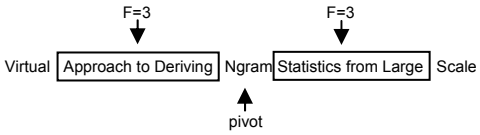


Figure 1: *2.F+1*-word size window context

Thus, as computation is involved, we need to process all possible substrings (continuous or discontinuous) that fit inside the window context and contain the pivot word. Any of these substrings is called a positional ngram. For instance, [Ngram Statistics] is a positional ngram as is the discontinuous sequence [Ngram ___ from] where the gap represented by the underline stands for any word occurring between Ngram and from (in this case, Statistics). More examples are given in Table 1.

Positional 2grams	Positional 3grams
[Ngram Statistics]	[Ngram Statistics from]
[Ngram ___ from]	[Ngram Statistics ___ Large]
[Ngram ___ ___ Large]	[Ngram ___ from Large]
[to ___ Ngram]	[to ___ Ngram ___ from]

Table 1: Possible positional ngrams

In order to compute all the positional ngrams of a corpus, we need to take into account all the words as possible pivot words. A simple way would be to shift the two-window context to the right so that each word would sequentially be processed. However, this would inevitably lead to duplications of positional ngrams. Instead, we propose a one-window context that shifts to the right along the corpus as illustrated in Figure 2. It is clear that the size of the new window should be *2.F+1*.

Figure 2: One-window context for *F=3*

This new representation implies new restrictions. While all combinations of words were valid positional ngrams in the two-window context, this is not true for a one-window context. Indeed, two restrictions must be observed.

Restriction 1: Any substring, in order to be valid, must contain the first word of the window context.

Restriction 2: For any continuous or discontinuous substring in the window context, by shifting the substring from left to right, excluding gaps and words on the right and inserting gaps on the left, so that there always exists a word in the central position *cpos* (Equation 2) of the window, there should be at least one shift that contains all the words of the substring in the context window.

$$cpos = \left\lfloor \frac{2.F+1}{2} \right\rfloor + 1$$

Equation 2: Central position of the window

For example, from the first case of Figure 2, the discontinuous sequence [A B _ _ E _ G] is not a positional ngram although it is a possible substring as it does not follow the second restriction. Indeed, whenever we try to align the sequence to the central position, at least one word is lost as shown in Table 2:

Possible shift	Central word	Disappearing words
[_ _ A B _ _ E]	B	G
[_ _ _ A B _ _]	A	E, G

Table 2: Shifting Substrings

In contrast, the sequence [A _ C _ E F _] is a positional ngram as the shift [_ A _ C _ E F], with C in the central position, includes all the words of the substring.

Basically, the first restriction aims at avoiding duplications and the second restriction simply guarantees that no substring that would not be computed in a two-window context is processed.

Virtual Representation

The representation of positional ngrams is an essential step towards efficient computation. For that purpose, we propose a reference representation rather than an explicit structure of each positional ngram. The idea is to adapt the *suffix* representation (Manber and Myers, 1990) to the positional ngram case.

Following the suffix representation, any continuous corpus substring is virtually represented by a single position of the corpus as illustrated in Figure 3. In fact, the substring is the sequence of words that goes from the word referred by the position till the end of the corpus.

Unfortunately, the suffix representation can not directly be extended to the specific case of positional ngrams. One main reason aims at this situation: a positional ngram may represent a discontinuous sequence of words. In order to overcome this situation, we propose a representation of positional ngrams based on **masks**.

[4] As specific, we intend a sequence that fits the definition of collocation given by (Dias, 2002): "A collocation is a recurrent sequence of words that co-occur together more than expected by chance in a given domain".

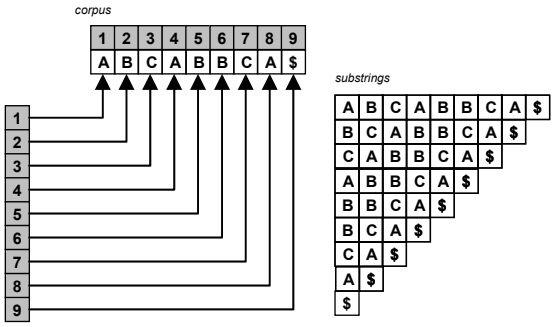


Figure 3: Suffix Representation[5]

As we saw in the previous section, the computation of all the positional ngrams is a repetitive process. For each word in the corpus, there exists an algorithmic pattern that identifies all the possible positional ngrams in a *2.F+1*-word size window context. So, what we need is a way to represent this pattern in an elegant and efficient way. One way is to use a set of masks that identify all the valid sequences of words in a given window context. Thus, each mask is nothing more than a sequence of 1 and 0 (where 1 stands for a word and 0 for a gap) that represents a specific positional ngram in the window context. An example is illustrated in Figure 4.

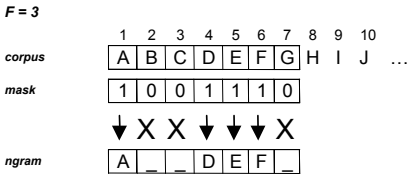


Figure 4: Masks

Computing all the masks is an easy and quick process. In our implementation, the generation of masks is done recursively and is negligible in terms of space and time. In Table 3, we give the number of masks *h(F)* for different values of *F*.

F	h(F)
1	4
2	11
3	43
4	171
5	683

Table 3: Number of masks

In order to identify each mask and to prepare the reference representation of positional ngrams, an array of masks is built as in Figure 5.

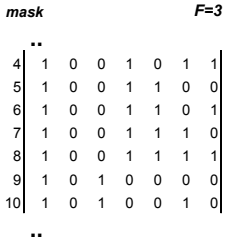


Figure 5: Masks Array

[5] The $ symbol stands for the end of the corpus.

From these structures, the virtual representation of any positional ngram is straightforward. Indeed, any positional ngram can be identified by a position in the corpus and a given mask. Taking into account that a corpus is a set of documents, any positional ngram can be represented by the tuple $\{\{id_{doc}, pos_{doc}\}, id_{mask}\}$ where id_{doc} stands for the document id of the corpus, pos_{doc} for a given position in the document and id_{mask} for a specific mask. An example is illustrated in Figure 6.

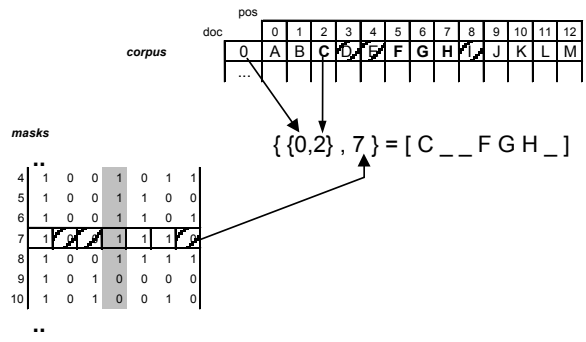


Figure 6: Virtual Representation

As we will see in the following section, this reference representation will allow us to follow the *Virtual Corpus* approach introduced by (Kit and Wilks, 1998) to compute ngram frequencies.

Computing Frequencies

With the *Virtual Corpus* approach, counting continuous substrings can easily and efficiently be achieved. After sorting the suffix-array data structure presented in Figure 3, the count of an ngram consisting of any n words in the corpus is simply the count of the number of adjacent indices that take the n words as prefix. We illustrate the *Virtual Corpus* approach in Figure 7.

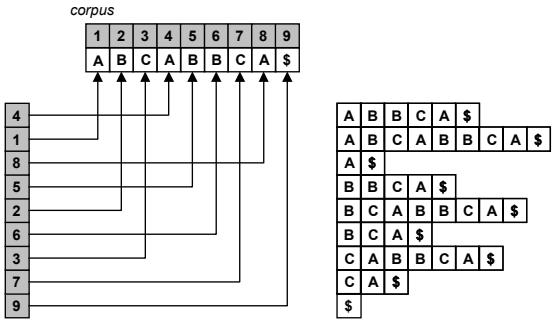


Figure 7: *Virtual Corpus* Approach

Counting positional ngrams can be computed exactly in the same way. The suffix-array structure is sorted using lexicographic ordering for each mask in the array of masks.

After sorting, the count of a positional ngram in the corpus is simply the count of adjacent indices that stand for the same sequence. We illustrate the *Virtual Corpus* approach for positional ngrams in Figure 8.

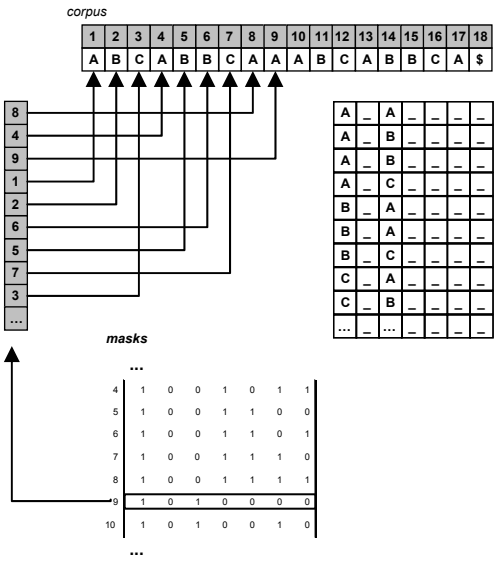


Figure 8: *Virtual Corpus* for positional ngrams

The efficiency of the counting mainly resides in the use of an adapted sort algorithm. For the specific case of positional ngrams, we have chosen to implement the multikey quicksort algorithm (Bentley and Sedgewick, 1997) that can be seen as a mixture of the Ternary-Split Quicksort (Bentley and McIlroy, 1993) and the MSD[6] radixsort (Anderson and Nilsson, 1998). Different reasons have lead to use the Multikey Quicksort algorithm. First, it performs independently from the vocabulary size. Second, it shows $O(N \log N)$ time complexity. Third, (Anderson and Nilsson, 1998) show that it performs better than the MSD radixsort and proves comparable results to their newly introduced forward radixsort.

The algorithm processes as follows: (1) the array of string is partitioned into three parts based on the first symbol of each string. In order to process the split, a pivot element is chosen just as in the classical quicksort giving rise to: one part with elements smaller than the pivot, one part with elements equal to the pivot and one part with elements larger than the pivot; (2) the smaller and the larger parts are recursively processed in exactly the same manner as the whole array; (3) the equal part is also sorted recursively but with partitioning starting from the second symbol of each string; (4) the process goes on recursively: each time an equal part is being processed, the considered position in each string is moved forward by one symbol.

As we already said, the efficiency of the counting mainly resides in the use of an adapted sort algorithm. Moreover, the sorting phase is the most time consuming of our global architecture that extracts collocations. So, we define a Parallel Sorting by Regular Sampling Multikey Quicksort algorithm in order to fasten this stage.

PSRS Multikey Quicksort Algorithm

The Parallel Algorithm we propose is based on Parallel Sorting by Regular Sampling (PSRS) as described in (Shi and Schaeffer, 1992). In particular, they apply their method to randomly generated 32 bit integers and use the classical quicksort (Hoare, 1962) as the sequential sorting algorithm. For a variety of shared and distributed memory architectures, their results display better than half linear speedups. Our PSRS algorithm sorts positional ngrams using the multikey quicksort as the sorting algorithm. Our algorithm can be divided into three distinct phases: a parallel multikey quicksort phase; reorganization by global pivots phase; a merge sort phase.

The parallel multikey quicksort phase consists in partitioning the original data (i.e. the corpus) into p contiguous lists, one per node[7], and uses the multikey quicksort algorithm to sort each local contiguous list. In fact, in step 1, each node reads the entire data file to one suffix array structure into his local memory as in figure 9[8].

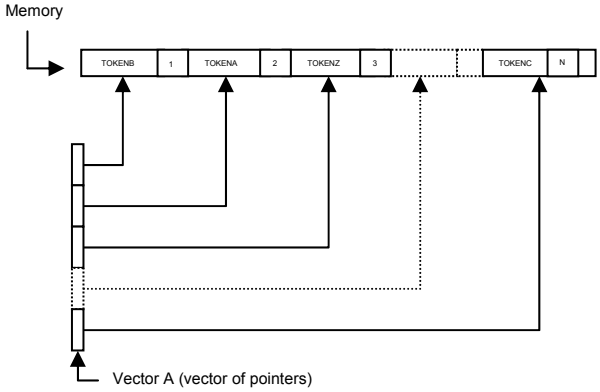


Figure 9: *Virtual Corpus* in memory

In step 2, each of the p nodes determines a contiguous list of size $w = N/p$ from the original data where N is the size of the corpus. In fact, each node makes a copy of size w of the suffix-array. We shall call this vector B which will be sorted. This situation can be seen in Figure 10.

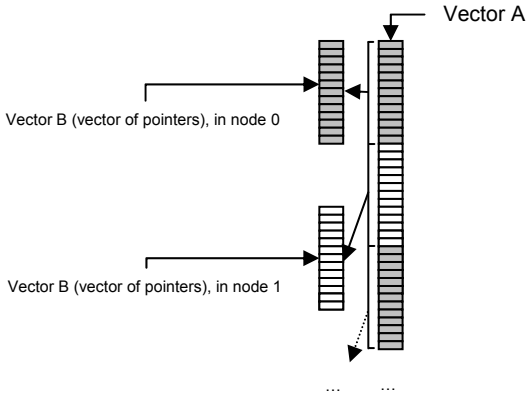


Figure 10: Suffix-array division

In step 3, each node sorts its local contiguous list using the multikey quicksort algorithm which implements the "*median of three modification method to improve the average performance of the algorithm while making the worst case unlikely to occur in practice*" (Lan and Mohamed, 1992). After this phase, all local contiguous

[6] MSD stands for Most Significant Digit.

[7] A node represents a processing unit.

[8] The reader will note that after each token we insert its position in the file.

lists are sorted following a given mask as shown in Figure 11.

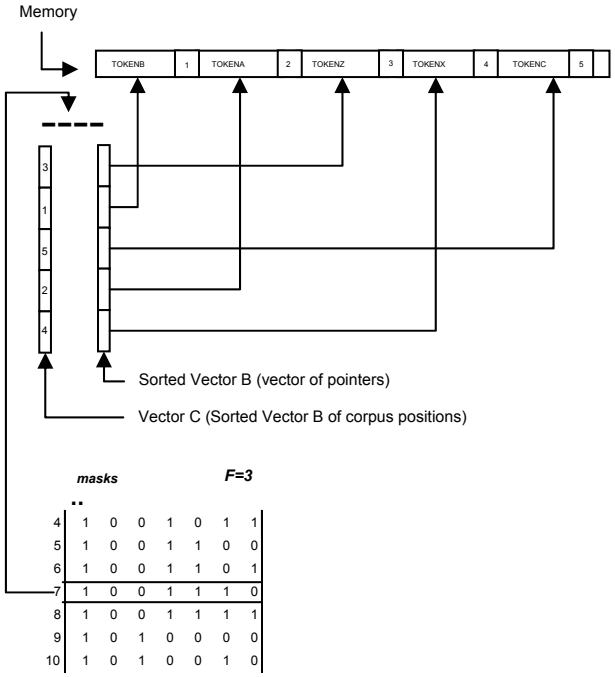


Figure 11: Sorted Suffix-array

The Vector C is constructed as, when we wish to communicate information with another node, we cannot exchange pointer information as they may not be the same on different nodes, but we can exchange information relative to the original positions of the tokens.

The reorganization by global pivots phase consists in (1) determining the *(p-1)* local pivots on each node, (2) determining the global pivots from the *p*(p-1)* local pivots and (3) reorganizing the local list in terms of the global pivots. The idea is to join and sort all local sorted contiguous lists in a parallel way with good load balancing. For that purpose, we use a Regular Sampling approach suggested by (Shi and Schaeffer, 1992) described as follows.

Each node determines *(p-1) local* pivots from its sorted list. The node 0 gathers the *p*(p-1)* local pivots (first inter-node communication). The node 0 calculates the global pivots from the list of all local pivots and broadcasts the global pivots to all nodes (second inter-node communication). This situation is shown in Figure 12.

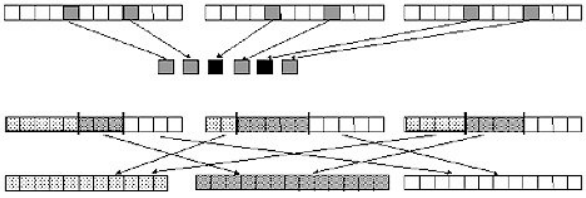


Figure 12: Reorganization for three nodes

Finally, the merge sort phase consists of creating on each node one final locally sorted list using a merge sort list. For that purpose, each node splits its sorted list into *p* sorted sub-lists based on the values of the global pivots.

Then, each node keeps one sorted sub-list and communicates the others to the appropriate nodes. Due to the fact that the sub-lists are of unequal size, we must first communicate the number of data items each node must send/receive from each other node before performing the actual data transfers (note that only vectors of integers will be passed i.e. integers representing the original token positions).

In order to reduce the communication costs and network traffic we use a customized collective communication (ALL-to-ALL) based on phases in which only pairs of processors communicate as shown in figure 13.

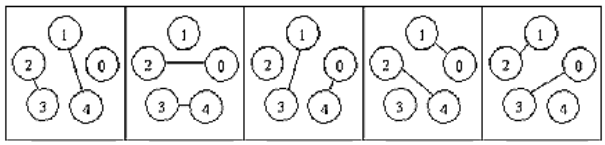


Figure 13: Collective communication

This step is then followed by a merge of the received vector[9]. In fact, each node merges the *p* sorted sub-lists into one local sorted list using the merge sort algorithm and then calculates the frequency of each ngram. Then, each node communicates the position of first instance of each n-grams plus its frequency to node zero where the matrix of all ngrams frequency is constructed.

The global architecture of our PSRS can be summarized as the following steps:

1. The original data file (size *N*) is copied to all *p* processor nodes of the cluster.
2. Each of the *p* nodes reads the data file to its local memory and builds the suffix-array.
3. Each of p nodes determines a contiguous list of size $w = N/p$ from the original data.
4. Each node creates the valid masks.
5. For each valid mask:
    a. Each node sorts the contiguous list using the multikey quicksort algorithm following the current mask.
    b. Each node determines *(p-1)* local pivots from its sorted list.
    c. The node 0 gathers all local pivots (first inter-node communication).
    d. The node 0 calculates the global pivots from the list of all local pivots.
    e. Node 0 broadcasts the global pivots to all nodes (second inter-node communication).
    f. Each node splits its sorted contiguous list into *p* sorted sub-lists based on the values of the global pivots.

[9] In particular, we can overlap communication and merge routines using the non-blocking send/receive routines of the MPI standard.

g. Each node keeps one sorted sub-list and passes the others to the appropriate nodes (third inter-node communication and the most expensive).
h. Each node merges the *p* sorted sub-lists into one local sorted list using the merge sort algorithm.
i. Each node communicates the position of first instance of each ngram plus its frequency to node zero.

6. The node 0 constructs the matrix of all ngrams frequency.

Results

The algorithm has been implemented using the single program multiple data (SPMD) programming methodology using the ANSI C programming together with the freely available MPICH implementation of the Message Passing Interface (MPI) library. A network of up to ten identical workstations was used. Each workstation consists of a single Pentium IV 2.4 GHZ processor with 512 Mb of RAM running the Windows XP operating system and they are connected via a 100 Mb Ethernet network.

We have conducted a series of experiments for various different sized sub-corpora of the CETEMPúblico Portuguese corpus using a seven-word size window context, for which we present two examples. The details of the two chosen test cases are given in Table 3.

Corpus	Case A	Case B
Size in Mb	6,829	20,477
# of words	1.000.000	3.000.000
#of ngrams	42.999.742	128.999.742

Table 3: Sub-Corpora Test Cases

The experimental results based on the two test cases are then presented in Tables 4 and 5.

Processors	Time/Minutes	Speedup	Efficiency
1	3,18		
2	2,11	1,51	0,75
3	1,79	1,78	0,59
4	1,46	2,17	0,54
5	1,32	2,42	0,48
6	1,18	2,69	0,45
7	1,15	2,77	0,40
8	1,03	3,10	0,39
9	1,00	3,16	0,35
10	0,98	3,24	0,32

Table 4: Results for the Sub-Corpora, Case A

In (Gil and Dias, 2002) the sequential algorithm took 8.34 minutes to sort and calculate the ngram frequencies of a 1.092.723-word corpus on an Intel Pentium III 900 MHz. Our result for a single processor for a 1.000.000 word corpus is 3.18 minutes which is to be expected as the performance of our single processor is approximately 2-3 times faster that the one used in (Gil and Dias, 2002). For Case B, a corpora three times that of case A, the time of 11,71 minutes is obtained for a single processor, in other words bearing out the *O(h(F) N log N)* complexity of the sequential algorithm.

Processors	Time/Minutes	Speedup	Efficiency
1	11,71		
2	7,64	1,53	0,77
3	5,79	2,02	0,67
4	4,79	2,44	0,61
5	4,22	2,78	0,56
6	3,91	3,00	0,50
7	3,68	3,19	0,46
8	3,50	3,34	0,42
9	3,34	3,50	0,39
10	3,27	3,58	0,36

Table 3: Results for the Sub-Corpora, Case B

The performance of the parallel algorithm is similar in both cases, slightly better in the larger case B as would be expected. Initially with a small number of processors reasonable speedups and efficiency are obtained but this parallel performance deteriorates with the augmenting number of processors due to high levels of communications. However the overall time taken for the sort is still a monotonically decreasing function when using up to 10 processors (See Figure 14 and Figure 15).

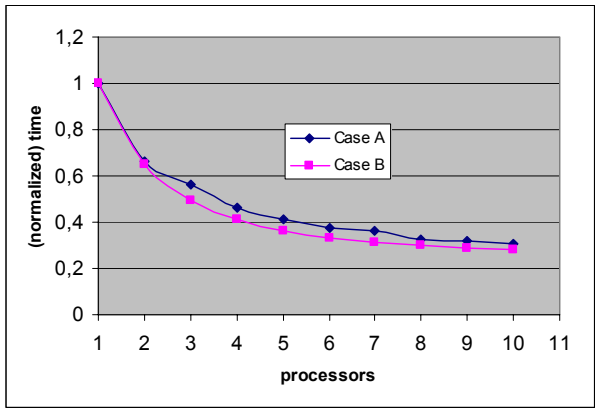


Figure 14: Running Time *vs* number of Processors

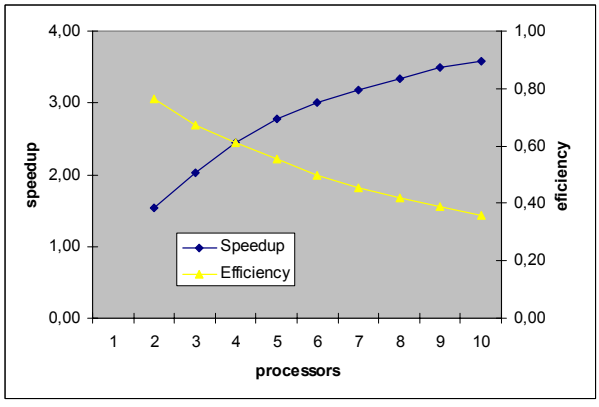


Figure 15: Speedup and Efficiency for Case B

Conclusions and Future Work

Mining multiword units in real life situation means the ability to deal with Gigabytes of data in a useful time frame necessitating the use of high performance computing architectures. Low cost network-based distributed and parallel architectures are a useful alternative for high cost proprietary machines (Kacsuk and Vajda, 1999) and offer a flexible, scaleable and readily available solution. In this paper, we have proposed a PSRS multikey quicksort algorithm to compute positional ngram frequencies which will be integrated to the SENTA system developed by (Dias, 2002). However, speedups can surely be achieved adapting to our current parallel architecture the work of (Yamamoto and Church, 2000) to positional ngrams that propose to compute categories of contiguous substrings instead of the substrings themselves.

References

Alexandre Gil and Gaël Dias. 2002. *Using Masks, Suffix Array-based Data Structures and Multidimensional Arrays to Compute Positional Ngram Statistics from Corpora*. Workshop on Multiword Expressions of the 41st ACL meeting. 7-12 July. Sapporo. Japan. http://www.di.ubi.pt/~ddg/publications/acl2003-1.pdf

Arne Anderson and Stefan Nilsson. 1998. *Implementing Radixsort*. ACM Journal of Experimental Algorithmics, Vol. 3. http://citeseer.nj.nec.com/79696.html

Chunyu Kit and Yorick Wilks. 1998. *The Virtual Approach to Deriving Ngram Statistics from Large Scale Corpora*. International Conference on Chinese Information Processing Conference, Beijing, China, 223-229. http://citeseer.nj.nec.com/kit98virtual.html.

Gaël Dias 2002. *Extraction Automatique d'Associations Lexicales à partir de Corpora*. PhD Thesis. New University of Lisbon (Portugal) and University of Orléans (France). http://www.di.ubi.pt/~ddg/publications/thesis.pdf.gz

Hoare, C. A. R. 1962. *Quicksort*. Computer Journal 5, 1(April 1962), 10-15.

John Sinclair. 1974. *English Lexical Collocations: A study in computational linguistics*. Singapore, reprinted as chapter 2 of Foley, J. A. (ed). 1996, John Sinclair on *Lexis and Lexicography*, Uni Press.

Jon Bentley and Robert Sedgewick. 1997. Fast Algorithms for Sorting and Searching Strings. 8th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orléans. http://citeseer.nj.nec.com/bentley97fast.html.

Jon Bentley and Douglas McIlroy. 1993. Engineering a sort function. *Software - Practice and Experience*, 23(11):1249-1265.

Kacsuk, P. and Vajda, F. 1999. *Network-based Distributed Computing*, Prospective Reports on ICST Research in Europe. http://www.ercim.org/publication/prosp/

Lan,Y. and Mohamed, M.A. 1992. *Parallel Quicksort in Hypercubes* Symposium on Applied Computing, Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's, Kansas City, United States, 1992, pp740-746.

Mikio Yamamoto and Kenneth Church. 2000. Using Suffix Arrays to Compute Term Frequency and Document Frequency for All Substrings in a corpus. *Association for Computational Linguistics*, 27(1):1-30. http://www.research.att.com/~kwc/CL_suffix_array.pdf

Message Passing Interface Forum. MPI: A message-Passing Interface Standard. Int. Journal of Supercomputer Applications, 8(3/4):165-414, 1994

MPICH MPI Software http://www-unix.mcs.anl.gov/mpi/mpich/

Shi, H. and Schaeffer, J. 1992. *Parallel Sorting by Regular Sampling,* Journal of Parallel and Distributed Computing, Vol 14, Nº 4, pp361-372.

Udi Manber and Gene Myers. 1990. *Suffix-arrays: A new method for on-line string searches*. First Annual ACM-SIAM Symposium on Discrete Algorithms. 319-327. http://www.cs.arizona.edu/people/udi/suffix.ps

Recognition and Parapharasing of Periphrastic and Overlapping Verb Phrases

Nobuhiro Kaji, Sadao Kurohashi

Graduate School of Information Science and Technology, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
{kaji,kuro}@kc.t.u-tokyo.ac.jp

Abstract

This paper proposes a dictionary-based method of recognizing and paraphrasing Japanese periphrastic and overlapping verb phrases. Periphrastic VP is a phrase in which the verb or the noun functions as an adverb, voice or aspect. Overlapping VP is a phrase in which there is an overlapping meaning between the verb and the noun. The result of our experiment showed that the method can deal with those two phrases effectively.

1. Introduction

A verb and a noun can form VP and usually convey the meaning in a compositional way. That is, the meaning of such VP is equal to the combination of the meaning of the verb and the noun. The verb represents an action and the noun represents its agent, object, etc.

However, VPs like (1a) and (2a) are different from such ordinary ones.

(1) a. to excel at teaching
b. to teach very well

(2) a. to ask a question
b. to ask something
c. to question

In (1a), it is not the verb 'excel' but the noun 'teaching' that represents an action, and the verb 'excel' functions as an adverb. Such VP is called *periphrastic verb phrase*. The meaning of (2a) is not equal to the combination of the meanings of the verb 'ask' and the noun 'question', because there is an overlapping meaning between them. Such VP is called *overlapping verb phrase*.

Periphrastic and overlapping VP can be paraphrased into more simplifeid expressions. For example, (1a) can be paraphrased into (1b), in which the verb 'teach' represents an action. (2a) can be paraphrased into (2b) or (2c), in which there is no overlapping meaning between the verb and the noun.

Paraphrasing periphrastic and overlapping VP is an important task, because of the following two reasons. Firstly, they can be paraphrased into more simplified expressions, and the simplification of text is useful for aphasic readers or deaf people etc (Carroll et al., 1999; Canning and Taito, 1998; Inui et al., 2003). Secondly, the identificaion of paraphrases enhances several NLP applications, such as information retrieval, question-answering, information extraction, and multi-documents summarization. In IR and QA, the precise treatment of paraphrases can raise the recall while keeping the precision (Jacquemin et al., 1997; Jacquemin, 1999; Lin and Pantel, 2001; Duclaye et al., 2003). In IE, a system uses patterns to capture events, and it is important to connect patterns that convey the almost same meaning (Shinyama et al., 2002; Shinyama and Sekine, 2003). In multi-documents summarization, it can avoid producing a repetitive summary (Barzilay et al., 1999).

However, paraphrasing periphrastic and overlapping VP has not been disucussed sufficiently. Although there are several related works that deal with paraphrasing periphrastic VP (Kozlowski et al., 2003; Furihata et al., 2004), the methods require hand-crafted semantic resource which is built by hand and not easy to prepare.

In contrast to those works, this paper proposes a method based on an ordinary dictionary, which is widely available these days. In this paper, we represent a dictionary-based metohd of recognizing periphrastic and overlapping VP. Futheremore, we show that the recognition of those VPs makes it possible to paraphrase them, from (1a) to (1b), and from (2a) to (2b) or (2c). Note that this paper deals with Japanese VP which consists of a verb and a noun.

The ramaining of this paper is organized as follows. In Section 2, we examine periphrastic and overlapping VP. In Section 3, an overview of our recognition metohd is shown, and the detail is described in Section 4 and 5. In Section 6, we show that the recognition of periphrastic and overlapping VP makes it possible to paraphrase them. The experimental results are shown in Section 7, and we conclude in Section 8.

2. Periphrastic and Overlapping VP

Periphrastic and overlapping VP are examined in this section.

2.1. Periphrastic VP

In periphrastic VP, one constituent represents an action and the other constituent functions as an adverb, voice or aspect. They are called *Main Constituent (MC)* and *Supplement Constituent (SC)*. Periphrastic VPs are classified into two types depending on which constituent is MC/SC. In the examples SC is underlined. [1].

Verbal-supplement-type A noun is MC and represents an action. A verb is SC and functions as an adverb, voice or aspect.

[1]Examples are Japanese and attached with English translation. In Japanese, a verb is located next to a noun, and a noun has postposition, such as '*-ga*' or '*-wo*' etc., which functions as a case maker.

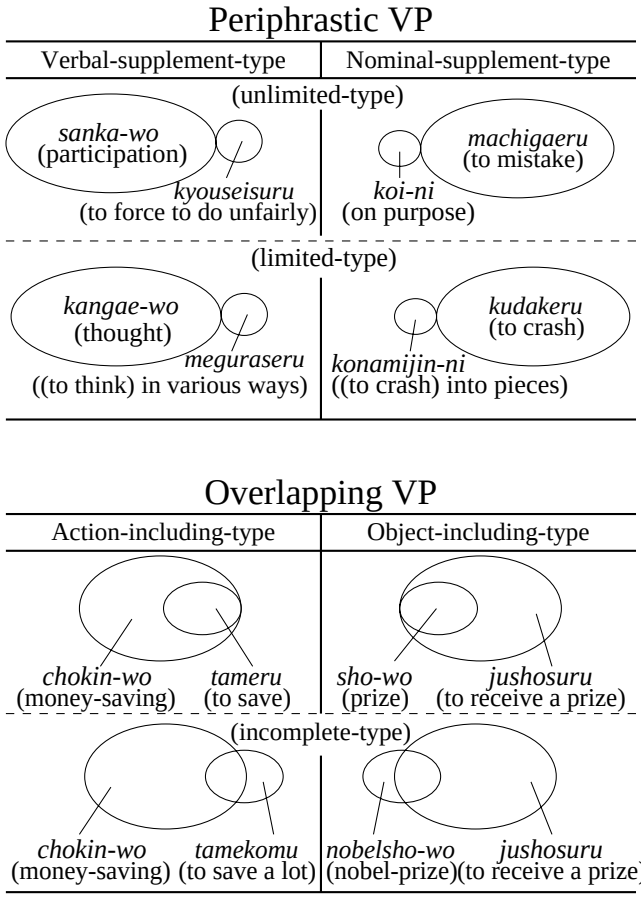


Figure 1: Classification of Periphrastic and Overlapping VP.

(3) a. *kaikaku-wo* *dankousuru*
reform to do resolutely
b. *sanka-wo* *kyouseisuru*
participation to force to do unfairly
c. *kangae-wo* *megurasu*
thought (to think) in various ways
d. *kaikaku-wo* *suru*
reform to do

In (3a) and (3c), the verb functions as an adverb. In (3b), the verb functions as an adverb and causative voice. In (3d), the verb '*suru* (to do)' is support verb. In this paper, support verb is regarded as one of the SCs.

Nominal-supplement-type A verb is MC and represents an action. A noun is SC and functions as an adverb, as in (4a) and (4b).

(4) a. *koi-ni* *machigaeru*
on purpose to mistake
b. *konamijin-ni* *kudakeru*
(to crash) into pieces to crash

In both types, some SCs can be connected with a large variety of MCs, and others can not. The fomer periphrastic VP is called *unlimited-type*, the latter is called *limited-type*. (3a), (3b), (3d) and (4a) are unlimited-type, and (3c) and (4b) are limited-type. For example, SC in (3a), '*dankousuru* (to do resolutely)', can be connected with a large variety of MCs such as 'discount', 'operation', etc. On the other hand, SC in (3b), '*megurasu* ((to think) in various ways)', can function as an adverb, only if it is connected with 'thought' or its synonyms.

Figure1 illustrates the classification of periphrastic VP. MC is represented by a large circle, and SC is represented by a small one.

2.2. Overlapping VP

In overlapping VP, the meaning of one constituent subsumes that of the other. The former is called *Main Constituent (MC)* and the latter *Included Constituent (IC)*. Overlapping VPs are classified into two types dpending on which constituent is MC/IC. In the examples MC is underlined.

Object-including-type The verb is MC and the noun is IC. The verb means 'an action and its object' and the noun means 'that object'. The meaning of the verb includes that of the noun, as in (5).

(5) *sho-wo* *jushosuru*
prize to receive a prize

Action-including-type The noun is MC and the verb is IC. The noun means 'an action and its object' and the verb means 'that action'. The meaning of the noun includes that of the verb, as in (6a) and (6b).

(6) a. *chokin-wo* *tameru*
money-saving to save
b. *uta-wo* *utau*
song to sing

In (6b), the meaning of the noun is approximately equal to that of the verb. Although such VP can be regarded as both object-including-type and action-including-type, we treat such VP as action-including-type.

In some overlapping VPs, the meaning of MC includes only part of the meaning of IC, as shown below.

(7) a. *nobel-sho-wo* *jushosuru*
nobel-prize to receive a prize
b. *chokin-wo* *tamekomu*
money-saving to save a lot

In (7a), the meaning of the verb includes a part of the meaning of the noun. Such overlapping VP is called *incomplete-type*. Similarly, (7b) is incomplete-type as well. Figure1 illustrates the classification of overlapping VP. The overlapping between the two circles represents the overlapping meaning.

It is worth pointing out that there are some VPs closely related to overlapping VPs.

(8) *sensei-ga* *oshieru*
teacher to teach

(8) is not overlapping VP because, strictly speaking, there is no overlapping meaning between the constituents. However, it is possible to regard that there is a kind of overlapping meanig, because the telic of 'teacher' is 'to teach'. There are several works addressing telic of nouns (Pustejovsky, 1991; Boni and Manadhar, 2002). Dealing with such VP is beyound the scope of this paper.

3. Overview of Recognition Method

Our dictionary-based recognition method uses definition sentences of constituents. An overview of our method is shown in this section. The detail is described in Section 4 and 5.

The input of our recognition method is VP consisting of a verb and a noun and a postposition. Given the input VP, the input VP and the definition sentences of constituents are processed by Japanese morphological analyzer (JUMAN)[2]and parser (KNP)[3]. If a constituent is polysemous and has more than one definition sentences, the appropriate one is selected by the word sense disambiguation process. After that, periphrastic and overlapping VP is recognized based on the definition sentences.

In what follows, the basic idea of our dictionary-based recognition and the method of word sense disambiguation are described.

3.1. Dictionary-based Recognition

Definition sentences of constituents can be used in order to recognize periphrastic and overlapping VP. The basic ideas are as follows:

- In unlimited-type periphrastic VP, the definition sentence of SC consists of (1) an adverb, (2) auxiliary verbs which function as voice or aspect, and (3) general VP such as 'to do something' or 'to do'. Such definition sentence can be detected by pattern matching rules. This method is called *pattern matching based recognition*.

- In limited-type periphrastic VP, a definition sentence of SC does not include general VP but contains the same word as MC with which SC can be connected. Therefore, limited-type periphrastic VP can be recognized by finding the alignment between MC and the definition sentence of SC. This method is called *alignment based recognition.*

- In overlapping VP, the definition sentence of MC includes the same word as IC. Therefore, overlapping VP can be recognized in the same way as limited-type periphrastic VP.

The detail of pattern matching based recognition and alignment based recognition is described in Section 4 and 5.

3.2. Word Sense Disambiguation

Dictionary-based recognitoin needs word sense disambiguation, because polysemous constituent has more than one definitions.

The disambiguation of the meaning of a noun requires information of wide context. But it is almost impossible to use enough information in our framework, since the input is VP consisting of a verb and a noun. Therefore, the input VP including polysemous noun is recognized as a periphrastic or overlapping, if one of its definition sentence fulfills a condition which is necessary for the input VP to be recognized as such.

On the other hand, in order to disambiguate the meaning of a verb, information of local context such as its argument is enough. The meaning of the verb is disambiguated using our method (Kaji et al., 2002). The method uses case frame dictionary acquired from large corpora automatically. Consider the following input VP:

- *oshie-wo* *aogu*
lesson 'to admire' or 'to require'

The verb '*aogu*' has two definition sentences: '(1) *uyamau* (to admire)' and '(2) *oshie-ya sashizu-wo motomeru* (to require lesson or instruction)'. The meaning of the verb '*aogu*' is disambiguated as follows:

- Select one case frame which is the most similar to the input VP. This case frame is called input case frame. In this example, the following input case frame is selected: {I, leader...}*ga* {expert, others...}*ni* {lesson }*wo aogu*.

- Extract case frames which are similar to the definition of '*aogu*'. They are called definition case frames.

- From definition case frames, select one case frame which is the most similar to the input case frame. In this example, the definition case frame of '*motomeru*

[2]http://www.kc.t.u-tokyo.ac.jp/nl-resource/juman-e.html
[3]http://www.kc.t.u-tokyo.ac.jp/nl-resource/knp-e.html

(to require)' is selected. As a result, it is said that the meaning of the '*aogu*' is '*motomeru* (to require)' which corresponds to the second definition sentence.

4. Pattern Matching based Recognition

SC is detected using its definition sentence and pattern matching rules. The recognition of SC makes it possible to recognize unlimited-type periphrastic VP.

4.1. Recognition of general VP

The definition sentence of SC consists of an adverb, auxiliary verbs, and general VP such as 'to do something' or 'to do' etc. In order to detect such definition sentence, it is necessary to recognize an adverb, auxiliary verbs, and general VP in the definition sentnece.

An adverb and auxiliary verbs are recognized by the result of morphological analysis. General VP is recognized by a simple rule. The general VP consists of:

- One or more nouns which represent general concept such as '*mono* (things)'. Such noun is called *Basic Noun (BN)*.
- One basic verb such as '*suru* (to do)'. Such verb is called *Basic Verb (BV)*.

Therefore, it is recognized by the list of BN and BV enumerated by hand. It is reasonable to think that the list has an enough coverage, because the words which are used in the definition sentences are restricted. The belows are parts of the lists:

Basic Nouns (BNs)	*nanika* (something) *koto* (thing) *mono* (something) *monogoto* (thing) ...
Basic Verbs (BVs)	*suru* (to do) *okonau* (to do) *dekiru* (to be able to do) *saseru* (to force to do) ...

Table 1: List of Basic Nouns (BNs) and Basic Verbs (BVs)

4.2. Unlimited-type verbal-supplement-type

The input VP is recognized as unlimited-type verbal-supplement-type, if the noun represents an action and the verb functions as SC.

The noun which has its verbal form can be considered to represent an action. Such noun can be detected by the result of morphological analysis.

If the verb is one of the BVs such as '*suro*(to do)', it is obvious that the verb functions as SC. Therefore, the verb is considered to function as SC, if:

- The verb is one of the BVs, or
- The definition sentence of the verb matches a regular expression pattern 'ADV∗ BN∗ BV AUX∗'. ADV and AUX mean an adverb and auxiliary verb respectively.

The belows are examples of the definition sentences recognized as that of SC:

kyouseisuru	*murini* unfairly:ADV	*sa* to do:BV	*seru* to force:AUX
dankousuru	*omoikitte* resolutely:ADV	*yaru* to do:BV	

4.3. Unlimited-type nominal-supplement-type

The input VP is recognized as unlimited-type nominal-supplement-type, if the noun functions as SC.

The recognition method of nominal-supplement-type is different from that of verbal-supplement-type, because of the following two reasons. Firstly, SC always functions as an adverb and does not function as voice. Therefore, its definition sentence always includes one or more adverbs, and does not includes an expression which represents voice. Secondly, the noun has a postposition, which functions as a casemarker. If only the noun has a postposition which functions as an optional casemarker, it can functions as SC.

Therefore, the noun is considered to function as SC, if:

- The definition sentence of the noun matches a regular expression pattern 'ADV+ BN∗ BV AUX∗'. Note that BV and AUX which function as passive or causative voice are not used, and
- The noun has a postposition '*-de*', '*-ni*' or '*-to*', which functions as an optional casemarker.

The belows are examples of the definition sentences recognized as that of SC:

koi-ni	*wazato* on purpose:ADV	*suru koto* to do:BV
tsuzukezama-ni	*tsugi-kara tsugi-ni* in succession:ADV	*suru koto* to do:BV

5. Alignment based Recognition

Limited-type periphrastic VP and overlapping VP are recognized by finding the alignment between one constituent and the definition sentence of the other.

5.1. Limited-type Periphrastic VP

Verbal-supplement-type The input VP is recognized as verbal-supplement-type, if the noun (=MC) represents an action, and the verbal form of the noun is equal to the head word of the definition sentence of the verb (=SC).

(9) a. *kangae-wo* *megurasu*
thought (to think) in various ways

b. *iroiroto* *kangaeru*
in various ways to think

(9a) is the input VP, and (9b) is the definition sentence of the verb in (9a). The head word of the definition sentence is 'to think', which is equal to the verbal form of the noun 'thought' in (9a).

Nominal-supplement-type The input VP is recognized as nominal-supplement-type, if:

- The verb (=MC) is equal to the head word of the definition sentence of the noun (=SC), and
- The noun has an optional casemarker, and
- The definition does not contain an expression that represents voice.

(10) a. *konamijin-ni* *kudakeru*
(to crash) into pieces to crash
b. *konagonani* *kudakeru*
into pieces to crash

(10a) is the input VP, and (10b) is the definition sentnece of the noun in (10a). The head word of the definition sentence is 'to crash', and it is equal to the verb in (10a).

5.2. Overlapping VP

Action-including-type The noun (=MC) means 'an action and its object', and the verb (=IC) means 'that action'. If the head word of the definition sentence of the noun is equal to the verb, the input VP is recognized as action-including-type.

(11) a. *chokin-wo* *tameru*
money-saving to save
b. *okane-wo* *tameru*
money to save

(11a) is the input VP, and (11b) is the definition sentnece of the noun. The head word of the definition sentence is 'to save', and it is equal to the verb in (11a).

Object-including-type The verb (=MC) means 'an action and its object', and the noun (=IC) means 'that object'. The head word of the definition sentence of the verb (=MC) is 'an action', and its argument is 'that object (=IC)'. Therefore, if the noun is equal to the argument in the definition of the verb, the input VP is recognized as object-including-type.

(12) a. *sho-wo* *jusho-suru*
prize to receive a prize
b. *sho-wo* *morau*
prize to receive

(12a) is the input VP, and (12b) is the definition sentence of the verb in (12a). The head word of the definition sentence is 'to receive'. Its argument is 'prize', which is equal to the noun in (12a).

5.3. Dealing with Mismatch Problem

In alignment based recognition, the definition sentence of one constituent is sometimes described using a synonym of the other. Especially in incomplete-type overlapping VP, a hypernym of IC is used. For example,

(13) a. *hitorigoto-wo* *iu*
talking to oneself to speak
b. *hitori-de* *hanasu*
to oneself to talk

(14) a. *Nobel-sho-wo* *jusho-suru*
Nobel-Prize to receive a prize
b. *sho-wo* *morau*
prize to recieve

(13a) is action-including-type overlapping VP, and (14a) is incomplete object-including-type overlapping VP. (13b) and (14b) are definition sentences of MC. (13b) is not described using IC 'to speak' but its synonym 'to talk'. (14b) is not described using IC 'Nobel-Prize' but its hypernym 'prize'.

This problem is solved using definition sentences, because a headword of a definition sentence corresponds to a synonym or hypernym of its entryword. The definition sentences of 'speak' and 'Nobel-Pirze' are as follows:

speak to talk

Nobel-Prize a prize given for important work in science, literature...

6. Paraphrasing

The recognition method makes it possible to paraphrase periphrastic and overlapping VP into simplified expressions.

6.1. Paraphrasing Periphrastic VP

Periphrastic VP can be paraphrased by transforming SC into adverbs or auxiliary verbs using its definition sentence.

(15) *sanka-wo* *kyouseisuru*
participation to force to do unfairly
→ *murini* *sankasa* *-seru*
unfairly to participate to force

kyouseisuru *murini* *sa* *seru*
unfairly to do to force

(16) *kangae-wo* *meguraseru*
thought (to think) in various ways
→ *iroiro* *kangaeru*
in various ways to think

megurasu *iroiroto* *kangaeru*
in various ways to think

In the examples, SC is underlined, and adverbs and auxiliary verbs into which SC is transformed are also underlined. In (15), SC is transformed into an adverb '*murini* (unfairly)' and an auxiliary verb '*-seru* (to force)'. MC 'participation' is transformed into its verbal form 'to participate'. In (16), SC is transformed into an adverb '*iroiro* (in various ways)'. MC 'thought' is transformed into 'to think'.

6.2. Paraphrasing Overlapping VP

Overlappping VP can be paraphrased in tow ways.

Elimination of IC Overlappping VP can be paraphrased by eliminating IC, from (17a) to (17b) and from (18a) to (18b).

(17) a. *sho-wo* *jusho-suru*
prize to receive a prize
b. *jusho-suru*
to receive a prize

(18) a. *chokin-wo* *tameru*
money-saving to save
b. *chokin-suru*
to save money

Transformation of MC After the elimination of IC, MC can be transformed using their definition sentences by Kaji et al.'s method, from (19a) to (19b) and from (20a) to (20b).

(19) a. *jusho-suru*
to receive a prize
b. *sho-wo* *morau*
prize to receive

(20) a. *chokin-suru*
to save money
b. *okane-wo* *tameru*
money to save

7. Experimental Results

The result of our method were evaluated by two judges: judge A and B. In the experiment, *reikai-shogaku-kokugojiten* dictionary was used(Tajika, 1997).

Recognition Experiment 600 VPs were extracted randomly from Mainichi news article corpus. Using those VPs, the recognition method was evaluated through the precision and recall. Table2 and Table3 show the evaluation by the judge A and B.

	Precision	Recall
Periphrastic VP	75%(42/56)	67%(42/63)
Overlapping VP	44%(4/9)	27%(4/15)
Total	71%(46/65)	59%(46/78)

Table 2: Recognition Result (Judge A)

	Precision	Recall
Periphrastic VP	64%(36/56)	63%(36/57)
Overlapping VP	22%(2/9)	18%(2/11)
Total	58%(38/65)	56%(38/68)

Table 3: Recognition Result (Judge B)

Paraphrasing Experiment 200 VPs which were recognized as periphrastic or overlapping VP by the recognition method were extracted randomly from Mainichi news article corpus. Those 200 VPs were paraphrased by the method, and the judges verified the result (Table4). The judge A verified that 163 of 200 were recognized correctly, and 147 of 163 were paraphrased correctly. The accuracy was 90%(147/163). According to the judge B, the accuracy was 92%(155/169).

	Accuracy	
	(Judge A)	(Judge B)
Recognition	82%(163/200)	85%(169/200)
Paraphrasing	90%(147/163)	92%(155/169)

Table 4: The Result of Paraphrasing Experiment

Examples Examples that were recognized and paraphrased successfully are shown below.

(21) a. *kakuho-ni* *tsutomeru*
maintenance to do hard
→ *isshokenmeini* *kakuho-suru*
hard to maintain
b. *isshokenmeini* *suru*
hard to do

(22) a. *chumoku-wo* *abiru*
watch to be done
→ *chumoku-sa-reru*
to be watched
b. *mizu-nado-wo* *kakeru*
water etc. to pour
c. *ukeru*
to be done

(23) a. *koso-wo* *matomeru*
working-out-idea to work out
→ *koso-suru*
to work out an idea
b. *kangae-wo* *matomeru*
idea to work out

(21a) was recognized as verbal-supplement-type, and paraphrased by transforming SC, '*tsutomeru* (to do hard)', into an adverb, '*isshokenmeini*(hard)'. (21b) is the definition sentence of SC '*tsutomeru*(to do hard)'. (22a) was also recognized as verbal-supplement-type, and paraphrased by transforming SC, '*abiru* (to be done)', into an auxiliary verb. SC '*abiru*' has two definitions: (22b) 'to pour water' and (22c) 'to be done', but the WSD method could selected the second definition successfully. (23a) was recognized as action-including-type, and paraphrased by eliminating IC, '*matomeru* (to work out)'. (23b) is the definition sentence of MC '*koso* (idea-working-out)'. It includes the same word as IC '*matomeru* (to work out)'.

Discussion The overall result is good, and indicates the effectiveness of our dictionary-based method. However, there are several problems.

Our method can deal with mismatch problems caused by synonyms or hypernyms. But, some mismatch problems requires more sophisticated method. For example,

(24) a. *koso-wo* *neru*
working-out-idea to think carefully
b. *kangae-wo* *matomeru*
idea to work out
c. *yoku* *kangaete* *yoimononisuru*
carefully to think to improve

(24a) is action-including-type overlapping VP. MC is the noun '*koso* (working-out-idea)'. (24b) is the definition of MC '*koso*', and it means 'to work out an idea'. (24c) is the definition of IC '*neru*', and it means 'to think care-

fully and improve'. It is difficult to recognize that these are overlapping between these definition sentences.

The result of overlapping VP recognition is not good. This is because some idiomatic VPs are wrongly recognized as overlapping VP.

(25) *iki-wo* *hikitoru*
breath to cease

(25) is an idiomatic VP that means 'to die'. One of the definition sentence of the verb '*hikitoru* (to cease)' is '*iki-ga taeru*', and contains the same word as the noun '*iki*'. Therefore, it is wrongly recognized as overlapping VP. Dealing with such idiomatic VPs is one of our future works.

8. Conclusion

This paper proposed the dictionary-based method of recognizing periphrastic and overlapping VPs. Futheremore, we showed that the recognition of those VPs makes it possible to paraphrase them. The experimental results indicated the effectiveness of the method. In a future work, we are going to apply the paraphrasing to NLP applications such as IR or QA etc.

9. References

Barzilay, Regina, Kathleen R. McKeown, and Michael Elhadad, 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for the Computational Linguistics (ACL99)*.

Boni, Marco De and Suresh Manadhar, 2002. Automated discovery of telic relations for wordnet. In *Proceedings of the 1st International WordNet Conference*.

Canning, Yvonne and John Taito, 1998. Syntactic simplification of newspaper text for aphasic readers. In *Proceedings of 22nd Annual International ACM SIGIR Conference*.

Carroll, John, Yvonne Canning, Guido Minne, Siobhan Devlin, Darren Pearce, and John Tait, 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL99)*.

Duclaye, Florence, FranÇois Yvon, and Oliver Collin, 2003. Learning paraphrases to improve a question-answering system. In *Proceedings of the 10th Conference of EACL Workshop Natural Language Processing for Question-Answering*.

Furihata, Kentaro, Atsushi Fujita, Kentaro Inui, Yuji Matsumoto, and Koichi Takeuchi, 2004. Paraphrasing of light-verb constructions based on lexical conceptual structure (in japanese). In *Proceedings of 10th Annual Meeting of the Association for Natural Language Processing (NLP2004)*.

Inui, Kentaro, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura, 2003. Text simplification for reading assistance: A project note. In *Proceedings of the Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP2003)*.

Jacquemin, Christian, 1999. Syntagmatic and paradigmatic representations of term variation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL99)*.

Jacquemin, Christian, Judith L. Klavans, and Evelyne Tzoukermann, 1997. Expansion of multi-word terms for indexing and retrieval using morphology and syntax. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL97)*.

Kaji, Nobuhiro, Daisuke Kawahara, Sadao Kurohashi, and Satoshi Sato, 2002. Verb paraphrase based on case frame alignment. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguisitics (ACL2002)*.

Kozlowski, Raymond, Kathleen F. McCoy, and K. Vijay-Shanker, 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexicgrammatical resources. In *Proceedings of the Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications(IWP2003)*.

Lin, Dekang and Patrick Pantel, 2001. Discovery of inference rules for question answering. *Journal of Natural Language Engneering*, 7(4):343–360.

Pustejovsky, James, 1991. The generative lexicon. *Computational Linguistics*, 17(4):401–441.

Shinyama, Yusuke and Satoshi Sekine, 2003. Paraphrase acuisition for information extraction. In *Proceedings of the Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications(IWP2003)*.

Shinyama, Yusuke, Satoshi Sekine, and Kiyoshi Sudo, 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the Human Language Technology Conference (HLT02)*.

Tajika, Junichi (ed.), 1997. *Reikai Shogaku Kokugojiten (Japanese dictionary for children)*. Sanseido.

Transducing Text to Multiword Units

C.H.A. Koster
Computing Science Institute,
University of Nijmegen,
The Netherlands,
E-mail: kees@cs.kun.nl

Abstract

In this working paper we discuss Head/Modifier trees and pairs as a representation for multi-word units (MWU's). We describe a system for the extraction of Head/Modifier trees (binary dependency trees) from running English text, and their unnesting to Head/Modifier pairs. The system is based on the EP4IR (English Phrases for IR) parser, which is freely available under the GPL licence. In the course of the parsing and transduction process, the system performs also the robust recognition of Named Entities and out-of-vocabulary wordforms, as well as a number of syntactic normalizations which serve to increase recall without impairing precision.

We give a taxonomy of HM pairs, based on the notion of aboutness, and describe the normalization, disambiguation and robustness techniques used in the EP4IR grammar.

We show that the Head/Modifier pairs generated by the parser/transducer can in their turn be used in a bootstrap process to improve the accuracy of the parsing process by resolving syntactic ambiguity. We discuss the problem of granularity in MWU's and conclude that HM trees are preferrable as a representation for MWU's. Finally we argue that novel WordNet-like resources should be developed, based on Head/Modifier trees.

The transduction of large volumes of text to linguistically meaningful Head/Modifier pairs provides a useful supplement to treebanks and a superior alternative to purely statistical co-occurrence-based techniques for obtaining Multi Word Units.

1 Introduction

This working paper is concerned with the extraction and use of a form of binary dependency trees as a representation for multi-word units (MWU's).

In section 2, we discuss different representations for linguistically motivated terms and argue that Head/Modifier pairs are a better representation for MWU's for applications in Linguistics and Information Retrieval than monolithic sequences or sets of words.

Section 3 describes the EP4IR (English Phrases for IR) system which extracts phrases from English text, transduces them to Head/Modifier trees and unnests them to Head/Modifier pairs.

In section 4 and the two following it we describe the strategies employed by the EP4IR system for the normalization of phrases, the reduction of ambiguity and for achieving robustness. These are crucial in assuring the quality of the generated HM trees.

In section 7 we discuss some applications of HM pairs as MWU's. We present some statistics on the HM pairs extracted from the OHSUMED 1987 collection and discuss the use of hyphenated forms as a source of collocations. We propose that the Head/Modifier pairs generated by the parser/transducer can in their turn be used in a bootstrap process to improve the accuracy of the parsing process and to resolve syntactic ambiguity.

Measurements given in section 7.1 show that, due to the use of Best-Only parsing, the speed of the system as developed for English is comparable to that of much shallower parsing techniques, and quite sufficient for even large-scale practical applications.

The transduction of large volumes of text to linguistically meaningful Head/Modifier pairs provides a useful supplement to treebanks and a superior alternative to purely statistical co-occurrence-based techniques for obtaining Multi Word Units. We argue that WordNet-like resources should be made to include Head/Modifier pairs or trees, because of their precision in resolving word sense problems and their value in paraphrasis.

The EP4IR grammar and its lexicon were developed in the IST-project Peking (see the (PEKING website)). The grammar is written in the AGFL formalism (Koster, 1991) which was developed for the syntactic description of natural languages and has a number of properties making it particularly useful for that purpose.

2 Phrases and Multi Word Units

The use of *Linguistically Motivated Terms* as indexing terms has always fascinated researchers in IR (for an overview see (Sparck Jones, 1999)), but the simple bag-of-words model with little linguistic support (stop-lists, lemmatization) still has not been outdone by other, more complicated document representations.

2.1 Phrases in Text Categorization

Text Categorization provides a good context for the study of phrasal document representations, because

it is easy to measure the effect of different representations on the categorization performance. There are many more phrases than words in a language, but luckily most categorization algorithms have a way to cope with large feature spaces.

Instead of words, non-linguistic representations like n-grams of consecutive characters have been investigated (Cavnar and Trenkle, 1994). But most research has addressed the use of phrases as terms, MultiWord Units extracted from the text. A distinction can be made between statistical phrases and syntactical phrases, depending on the way in which they are identified in the text. A spectrum of techniques for phrase extraction has been used:

- statistical phrases, chunks or collocations: sequences of k non-stopwords occurring consecutively (Cohen and Singer, 1996) or even taken from anywhere in the document (Caropreso et al, 2000)
- syntactical phrases identified by shallow parsing (Alonso and Vilares, 2002), template matching or finite state techniques (Grefenstette, 1996)
- Head/Modifier pairs obtained by shallow parsing (Lewis, 1992) or deep parsing (Fagan, 1988; Strzalkowski, 1995).

It is a disappointing fact that over the years none of the experiments using any sort of statistical or linguistical MWU's for categorization has shown a marked improvement over the use of single keywords.

Our own experiments in Text Categorization with linguistically motivated MWU's (Bel et al, 2003) and HM pairs (Koster and Seutter, 2003) also failed to produce a worthwile improvement over keywords, in spite of the heavy investment in linguistical resources.

2.2 Structured Head/Modifier pairs as MWU's

In all of the previous approaches, including our own experiments, a phrase like software engineering was considered as a monolithic multiword unit, and no use was made of its inner structure. Head/Modifier pairs however have a structure, the polarity between head and modifier, which researchers like (Caropreso et al, 2000) explicitly eliminate by sorting the elements, treating the pair like a set instead of a sequence. But that may eliminate differences like that between the management of science and the sciencee of management.

Apart from the conflation of equivalent terms, for which HM pairs give more scope because head and modifier can be independently conflated, it should be pointed out that there is more information in the HM pair engineering, software than in the collocation software engineering: for computing the relevance of a term to a class c we can use $P(c|head, modifier)$ in both cases, but $P(c|head, \overline{modifier})$ only for the HM pair – in this case the information that the document is about *another* form of engineering.

Instead of (unstructured) collocations we propose to use (structured) Head/Modifier pairs for representing MWU's.

3 HM trees and HM pairs

We now introduce the EP4IR system for obtaining HM pairs from English text, which performs the following process:

1. parsing the text by means of a robust rule-based Best-Only parser
2. transducing the phrases found in it to Head/Modifier trees
3. unnesting the trees to Head/Modifier pairs.

In the course of the parsing and transduction process, the system performs also the robust recognition of Named Entities and out-of-vocabulary wordforms, as well as a number of aboutness-preserving syntactic normalizations (word order normalization; elision of mood, time and modality; passive-to-active transformation) which serve to increase recall without impairing precision.

These transformations, as well as the (compositional) transduction of phrases are described in the grammar along with the structure of the phrases.

3.1 HM trees

The *Head/Modifier trees* produced by the transduction are binary dependency trees, structures of the form

[head, modifier]

where both the head and modifier are either (possibly empty sequences of) words extracted from the text or, recursively, other HM trees.

In the course of the transduction, elements which are not considered meaningful for the *aboutness* of the text (Bruza and Huibers, 2004) are rigorously elided: articles, quantifiers, determiners and adverbs (which according to (Arampatzis et al., 2000) contribute little to the aboutness of a phrase).

The elements (heads and modifiers) of a HM tree are prefixed with lexical types (N: for noun, A: for adjective, V: for verb form; also X: for adverb Q: for quantity and P: for pronoun). These prefixes, which can be cumulated (see 5.1), can be used by the lemmatizer and may be removed by untyping.

3.2 HM pairs and unnesting

The intuition behind a HM pair `[head, modifier]` is that the modifier is joined to the head to make it more precise, i.e. in order to distinguish between various meanings of a polysemous head. The modifier may also be empty.

The HM trees are in fact recursively composed out of HM pairs, like

```
[[ N:mower, N:lawn], A:large]
[ N:operation, [V:consuming, N:time]]
```

head	modifier				
	V	N	P	A	PP
V	–	object relation		–	yes
N, P	subject relation	predicative/attributive relation			yes
A	–	–	–	–	yes

Figure 1: Relations realized by HM pairs

Any HM tree F may be unnested into a set of HM pairs S by repeatedly taking some pair without nesting from F, replacing it by its head, and adding that pair to the set S, until F is empty. Some (artificial) examples:

```
[a, b] ==> { [a, b] }
[a, [b, c]] ==> { [b, c], [a, b] }
[[a, b], c] ==> { [a, b], [a, c] }
```

This process may optionally include morphological normalization and removal of the types:

```
[[N:mower,N:lawn],A:large] ==>
{[mower,lawn],[mower,large]}

[N:operation,[V:consuming,N:time]] ==>
{[operation,consume],[consume,time]}
```

We allow the |-sign as an or-operator:

```
[a, b | c ] ==> { [a, b], [a, c] }
[a | b, c ] ==> { [a, c], [b, c] }
```

As an example, the structure of common SVOC sentences may be schematically expressed by

```
[subject,[verb,object|complements]]
```

which is unnested to

```
{[subject,verb][verb,object]
  [verb,complements]}
```

A tree with curly outer brackets instead of square brackets, typically occurring nested in another one, is unnested with an empty abstraction. The sentence I saw the man whom you gave a book produces the tree

```
{P:I,[V:saw,N:man{P;you,[V:gave,N:book|to
  N:man]}]
```

which is unnested, lemmatized and untyped to

```
{[I,see],[see,man],[you,give],[give,book]
  [give,to man]}
```

Verb/verb pairs are avoided: I prefer to read a book yields the tree

```
[P;I,[V:prefer|[V:read,N:book]]
```

which yields the pairs

```
{[I,prefer][I,read][read,book]}
```

3.3 A Taxonomy of HM pairs

There are four main forms of pairs, corresponding to four important syntactic (and indirectly semantical) relationships:

1. the *subject relation* between the head `N` of an NP and the main verb `V` governed by it is expressed by a pair `[N,V]`
2. the *object relation* between a (transitive) main verb `V` and its object `N` is expressed by a pair `[V,N]`
3. the *predicative* relation between a subject `N` and the predicate `P` is expressed by a pair `[N,P]`; a similar pair `[N,A]` results from the *attributive relation* between a noun `N` and an adjective `A`. As an example, both the car is red and the red car will yield a pairs `[car,red]`
4. the *prepositional relation* between the head of a verb phrase or noun phrase and a PP modifying it, e.g. `[see,with telescope]` or `[man,with telescope]`.

The *prepositional relation* is expressed by transducing the preposition complement as an additional modifier (prefixed by `|`). The example sentences

I gave the trip to Paris to Paris

I was seen by him by the tree

are transduced to

```
{P:I,[V:gave,[N:trip,|to N:Paris]|to N:Paris]}
{P:him,[V:seen,P:I|by N:tree]}
```

which is unnested, lemmatized and untyped to

```
{[I,give],[give,trip],[trip,to Paris],
   [give,to Paris]}
{[he,see],[see,me],[see,by tree]}
```

Other relationships (negation, auxiliary verbs, adverbial modifiers) are presently not expressed in the transduction, even though they are recognized by the grammar and could easily be added.

4 Normalization in the EP4IR grammar

In this section we describe the phrase normalization techniques embodied in the EP4IR grammar.

4.1 Morphological normalization

All words or collocations occurring in the HM pairs may optionally be morphologically normalized by lemmatization, in order to conflate different forms. The lemmatization process is aided by the grammar, which prefixes each normalizable element with its class.

As an example, both the predicative relation `[N:man, A:V:sneezing]` obtained from the sneezing man and the subject relation `[N:man, [V:sneezed,]]` obtained from this man sneezed will by unnesting, lemmatization and untyping lead to `[man, sneeze]`.

4.2 Syntactical normalization

For each construct described by the grammar, its transduction is included in the rule describing the construct. Thus, the transductions of complicated constructs are expressed compositionally in terms of those of their components. In the process, elements may be elided or re-ordered and additional symbols injected (like the `[`, `,` and `]`) in order to express uniformly the relations described above.

The syntactic normalizations implemented in this way include *de-passivation*: the sentence

the train was driven by a clockwork engine

is, by unnesting and morphological normalization, turned into

`[engine, clockwork]` `[engine, drive]` `[drive, train]`

4.3 Spelling normalization

Some words and some collocations (well known, well-known, wellknown) can be written in several ways, which have to be mapped onto one same spelling. As an example, the word for Central Processing Unit may variously be spelt as cpu, CPU and c.p.u, apart from being written out in full as a collocation. In order to bring all these variants together, words and collocations in the lexicon may be equipped with a unique transduction, in all these cases CPU.

The same mechanism may be used to deal with abbreviations and variant spellings (like leave for the noun leaf, not to mention British vs American spelling and typical errors like parralel for parallel). In the current lexicon, this mechanism has been used only sparingly.

A related problem is that of *de-capitalization*. The use of capital letters may provide essential information (e.g. in recognizing personal names) which is lost if the text is simply de-capitalized. The strategy followed in AGFL and therefore EP4IR is to take capital letters in the lexicon literally, but to allow a capital letter in the text to stand for a small letter in the lexicon. This is based on the observation that there are many reasons for capitalizing a small letter (first word of the sentence, initial letters of words in titles of publications, or just personal taste of the author), whereas there are no general reasons for decapitalizing a capital letter.

5 Disambiguation techniques

Ambiguity is the bane of Natural Language Processing. In our case, the parser should not find all analyses, but rather a single most probable one. To that end, parsings are ranked by their number of *penalties*, and the parser takes the/a solution with the lowest number of penalties.

The EP4IR grammar includes a number of techniques to reduce ambiguity, aimed at either syntactic or lexical ambiguity. The lexicon used by EP4IR is mostly based on the WordNet 1.6 lexicon (Miller et al., 1993), using additional lexical material from various sources. WordNet is notoriously unhelpful in assigning word classes, being much too permissive ("if I can use this word as a noun in some sentence then, by golly, it must be a noun"). The lexicon and the grammar had to be adapted to one another in order to eliminate spurious ambiguities.

5.1 Lexical disambiguation

In English, and especially in WordNet, many words can belong to more than one syntactical category, like the word only (adverb, adjective, possibly noun). In order to reduce the degree of lexical ambiguity, an attempt is made to reduce the number of categories per word to one wherever possible, using syntactic promotion rules.

5.1.1 Syntactic promotion

In the EP4IR lexicon words obtain as far as possible only one word class, but certain *promotion rules* are present in the grammar, allowing the acceptance of a word from one lexical category for another category at the price of a penalty:

- (A:→N:)
  since syntactically any adjective can be used as the head of a noun phrase, there is no sense in including any adjective explicitly amongst the nouns
- (V:→A:)
  many adjectives are in fact participles (even though there may be some semantic difference, as in drunk), and the grammar will allow such verb forms at adjective positions, as in the working man. Therefore all present and past participles have been removed from the adjectives. As a consequence, the grammar considers the progressive form (he is asking the president for clemency) as a predicative sentence.

In Wordnet, many words occur both as an adverb and as an adjective. At predicative positions it hardly matters whether the word is accepted as an adverb or as an adjective. Consequently, adjectives and adverbs that can not be used attributively have been marked, and at predicative positions the two are not distinguished.

In Wordnet, all particles occurring in phrasal verbs like to go back or to put up (with something) have also been included as adverbs. Since the EP4IR grammar includes a detailed treatment of phrasal

verbs, those particles (mainly prepositions) have been removed from the adverb category. Also the prepositions have been removed from the adjectives. The ordinal and cardinal numerals have been removed from the nouns, adjectives and adverbs, but they may be accepted as such by promotion of numberals to adjectives.

5.2 Syntactic disambiguation

The two main sources of structural ambiguity in English (apart from the lexical ambiguities, mentioned above) are *NP substructure* and *PP attachment*. In order to obtain reliable HM pairs, ambiguity has to be avoided, but reliable HM pairs can also be used to reduce ambiguity.

5.2.1 Disambiguating the NP substructure

In English, any sequence of more than two nouns and/or adjectives is ambiguous, since the rule for noun part is both left- and right-recursive. Very schematically:

```
noun phrase:
  [determiner], noun part,
      (connector, noun phrase;  ).
noun part:
  adjective, nounpart;
  noun part, noun part.
```

In EP4IR the rule for `noun part` is only right-recursive, allowing left-modification just by a single noun

```
noun part:
  adjective, nounpart;
  noun, noun part.
```

but this noun may well be a collocation like trade union, which makes the 3-noun NP trade union leader unambiguous. Similarly, the collocations off duty and night nurse disambiguate an off duty night nurse. Collocations (or MWU's) may play an important role in disambiguating NP substructure (but see 7.4). The EP4IR lexicon contains a total of 485 MW adjectives and 78350 MW nouns, gleaned from Wordnet or derived from the EPO1F corpus (see section 7.3).

There is also a rudimentary treatment of NP-like time and place expressions (like this morning) in order to avoid their analysis as NP's and therefore as complements to the verb part.

5.2.2 Disambiguating PP attachment

In English, any sequence of two or more Preposition Phrases is ambiguous, since the second PP may either belong to the head of the first PP or to the same head as the first PP. In EP4IR subcategorization information provided by the lexicon is used to preferentially attach certain PP's to the verb part (as in to compare NP with NP or the instrumental use of with). In particular, the passive form has a strong affinity for by NP. Of course the (in)transitivity of the verb is used to determine the possible presence of the object, but in English a potential object may still be absent (I shoot you; I shoot).

Using subcategorization information about adjectives, PPs belonging to the adjective (e.g. reminiscent of NP) are also suitably attached. PP's with the preposition of are preferentially attached to the preceding NP (but see the following section). PP's following the NP may be attached to it in various dependency structures. This form of ambiguity is not satisfactorily resolved for lack of information about the subcategorization of the noun.

5.3 Collocations

By a *collocation* we shall mean a sequence of two or more words that preferentially occur together (more than is likely by chance). Additionally, the collocation may have a *non-compositional meaning* (in combination the words mean something else than is to be expected from their individual meaning) and *frozen syntax*, in the sense that syntactic variation (e.g., substitution of other words) is limited.

It makes sense to include collocations like software engineering as one word in the lexicon, in order to assure that modern software engineering is not interpreted as the engineering of modern software. WordNet 1.6 contains a rather haphazard collection of collocations, and special domains should have their own special collocations. In the lexicon, collocations may include a transduction to the same pair that they would obtain in isolation, e.g. `[engineering,software]`, in order to ensure that they will be conflated with their syntactic variants like the engineering of software.

6 Robustness measures

Under this heading we mention the measures introduced to cope with the following problems:

1. words that do not occur in the lexicon
2. unknown or incorrect syntactic constructions.

6.1 Dealing with unknown words

The parser will be faced with many words (about 10 to 20% in typical IR applications) that are not in the lexicon, for different reasons:

- names
  any text will introduce personal names or names of companies that can not usefully be foreseen in the lexicon. The EP4IR lexicon contains the most frequent English christian names, as well as some haphazard names of historical Americans gleaned from Wordnet. As a robustness device, it will also recognize words starting with a capital letter and partly or wholly capitalized as names, and therefore nouns. There should be some sub-grammar around it to deal properly with titles and modes of addressing (the third Secretary of the Central Committee of the German Democratic Republic). The same holds, of course, for names of companies.

- technical terms
  In a text on chemistry or medicine, many specialized words will occur that warrant the use of special domain lexica. From WordNet and other sources, the EP4IR lexicon contains a plethora of chemical and biological words and collocations. For other domains, it is relatively easy to find most of the domain words by filtering the unknown words from a large corpus. However, developing this word list into a full-fledged domain lexicon (including collocations) is a major task.

- newly created words
  Through ignorance or creativity, people are always inventing new words, mostly by compounding known words ("Z-beam"; "3-phase-constriction") or by derivation from known words ("downwardly"; in one patent application even "downwardlyly").

  A number of robust rules for recognizing unknown words with typical noun-, adjective- and verb-endings has been included, which is especially effective in the case of medical and chemical texts. The same mechanism is of help in the case of new words arising through ignorance or error (e.g. in English patent applications poorly translated from a Japanese original).

  The approximate matching of unknown words against known words from the lexicon might deal with many spelling errors, but this last mechanism has not been implemented yet.

- choosing between parsings
  In EP4IR, use is made of *penalties* to penalize unlikely analyses: from left to right the longest segment with the lowest number of penalties is taken. Of course these preferences are at best justifiable heuristics; much better behaviour is to be expected from a probabilistic parser taking syntactical and lexical frequencies into account. With the present state-of-the-art in semantics, very little help can be expected from semantical analysis.

6.2 Dealing with incorrect syntax

Conventional linguistic parsers serve to find all syntactically correct analyses for each sentence of input. The parser is presented with a corpus nicely segmented into sentences, one per line. In the IR situation however, the goal is to extract as much information as possible from the running input. The input consists of multi-sentence paragraphs, which are hard to segment into sentences automatically.

The parsing strategy followed in EP4IR is *segment parsing*, analysing from left to right, skipping unknown words between segments and accepting not only complete sentences, but, failing that, also other recognizable fragments like a noun phrase. Furthermore, the parser can relax the strict agreement rules where necessary.

7 HM pairs as MWU's

In this section we discuss some of the issues in using HM pairs as MWU's. First we give some statistics and measurements concerning the extraction of HN pairs from a large corpus. Then we discuss the bootstrapping of MWU's to derive more MWU's and the mining of hyphenated form to obtain MWU's. We discuss the problems of granularity, which HM pairs have in common with more traditional MWU's and propose to use HM trees to represent MWU's.

7.1 The OHSUMED corpus

We have analyzed all medical abstracts from 1987 in the OHSUMED corpus, taking only the narrative parts, and transduced and unnested them to HM pairs, omitting those with an empty modifier. This process took 11hrs 45 minutes, 42300 seconds in total, on a 700Mhz INTEL PC. The following table shows some statistics of the resulting pairs:

5841173	words in the corpus
2930384	HM pairs resulting
1603562	different HM pairs, of which
321772	non-hapax.

Like in earlier studies (Koster and Seutter, 2003), about 80% of the HM phrases are hapaxes. The parse time can be reduced by taking a faster computer (my laptop is four times as fast), or using computers in parallel.

The distribution of the types of the resulting HM pairs is as follows:

head	modifier				
	V	N	P	A	PP
V	5194	294573	22246		291320
N	248815	690470	24755	711455	249184
P	143335				8714
A					3602

7.2 Bootstrapping MWU's

(Bolshakov, 2004) defines a *collocation* as

> a syntactically connected and semantically compatible pair of content words.

In this sense, we are only interested in those MWU's that are collocations. Although HM pairs, unlike statistical phrases are syntactically derived and therefore syntactically compatible, they still need human scrutiny in order to avoid errors and verify semantical compatibility. This problem is to a large extent caused by unresolved ambiguity, rather than by the text containing gibberish.

Any unambiguous NP consisting of two elements (nouns, adjectives, also numerals etc) can be accepted as a collocation (especially if it is frequent) and added to the lexicon, after which it will disambiguate all NP's of three elements in which it occurs, and even some NP's of four elements. This *bootstrap* approach to collocations can save a lot of manual work.

7.3 Mining hyphenated forms

The mining of hyphenated forms in a text presents a novel technique for deriving MWU's. By a *hyphenated form* we mean a sequence of words joined together by hyphens, such as the term Multi-Word-Unit itself. Apart from some specialized technical use of hyphenation (see later), they often have the form of Noun Phrases, and syntactically occur as modifiers a larger NP. This also explains *why* their author thought it necessary to use hyphens instead of spaces: in order to avoid ambiguity. The author went out of his way to ensure a particular reading of the text.

In order to investigate their nature and prevalence, we have extracted all hyphenated forms from the EPO1F corpus (described in (Koster and Seutter, 2003)), which consists of 16000 Patent Applications written in English, obtained from the European Patent Office. The corpus is about 73 Million words long. For the extraction we used a much simplified NP grammar with hyphens as word separators.

The 73 Million words included 338261 hyphenated forms. Rejecting the ones that did not contain at least one word from the English lexicon left 320112 occurrences of 39474 different hyphenated forms. Some examples clearly show the attempt to construct a modifier for which a single word is not available: second-from-the-right, state-of-the-art, bread-and-butter, signal-to-noise, L-shaped, cut-off, multi-unit.

The length distribution of the hyphenated forms was as follows:

number of elements	number of occurrences	different elements
2	305619	36026
3	12891	2725
4	14110	3248
5	1437	644
6	112	63
7	15	9
> 7	38	16

The longest linguistically meaningful hyphenated form found in the corpus was detected-level-comparison-and-discrimination-by-denomination, which makes little sense without its context. Among the longest forms were found many chemical formulae like benzyl-dimethyl-2-hydroxy-ethyl-ammonium-chloride and amino sequences like gly-glu-phe-tyr-phe-asp-leu-arg-leu-lys-gly-asp-lys which deserve a separate treatment.

Hyphenated forms are a rich source of well-attested Multi-Word-Units.

7.4 The problem of granularity

Upon inspection of the HM pairs generated from the OHSUMED corpus, numerous examples were found of a noun that occurred both in some collocation (from the EP4IR lexicon) and as the head of some HM pair, e.g.

blood_flow	[flow, coronary]
heart_disease	[disease, bone]
malignant_tumor	[tumor, mammary]

Therefore the knowledge is lost that both blood flow and coronary flow are an example of flow. For collocations with a completely non-compositional meaning this is is correct, but for the frequent compositional collocations the loss of compositional structure is serious (a blood flow meter should be an example of a flow meter). This problem is present in all monolithical MWU representations (those without internal structure). A similar problem occurs in languages with compound nouns, like Dutch and German, where it was found that adding the elements of compound nouns to a query enhanced recall (Kraaij and Pohlmann, 1998).

A possible solution is used by EP4IR: to transduce a non-compositional collocation as a monolithical multiword unit and compositional ones to structured MWU's (HM pairs and trees).

But the question also arizes whether it is always wise to unnest HM trees into pairs. The presence of many collocations consisting of more than two elements shows that it is necessary to use HM trees directly as MWU's and as indexing terms for retrieval and categorization.

7.5 HM-based linguistic resources

WordNet, although it is about words, contains a striking number of multi-word units, as periphrastic terms in synsets, or as short descriptions. In applying the WordNet concept to other languages, especially in EuroWordNet, it was often found that a concept denoted by one word in one language could only be paraphrased as a MWU in another. HM pairs (or in some cases shallow HM trees) would form a much better basis for a WordNet than single words, because they are more precise, and the problem of the unspeakable word is largely solved. But HM trees would be even better, because they provide a solution to the granularity problem.

8 Availability

The 'English Phrases for IR' (EP4IR) grammar can be downloaded from the (AGFL website), together with its lexicon and the AGFL parser generator system, which is available under the GPL/LGPL licence (see (Koster and Verbruggen, 2002)).

9 Conclusions

We have in this working paper investigated the production of HM pairs and trees, and their use as MWU's. We gave some arguments showing that such structured representations of MWU's are preferable to monolithical representations. The problem of granularity was raised, and we conclude that HM trees may be the best representation of MWU's for Information Retrieval and terminology building.

Applied linguistics could benefit tremendously from the availability of linguistic resources based on HM trees rather than words. As shown above, parsing could become more precise when using pair occurrence frequencies. The use of HM-based terms could solve many sense disambiguation problems in Information Retrieval. Crosslingual Retrieval and Categorization would benefit enormously from monolingual and multilingual HM-based lexica including lexico-semantical relations. Even machine translation might get a fresh impulse from such resources.

Many monolingual and multi-lingual already resources developed for machine translation, embodying detailed linguistic knowledge, could be used to derive such resources. And the techniques described in this paper can be used to generate vast numbers of HM trees from available texts and the internet.

References

AGFL website: `http://www.cs.kun.nl/agfl/`

Steven P. Abney. Parsing by Chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pp 257–278. Kluwer Academic Publishers, Boston, 1991.

A. Arampatzis, Th.P. van der Weide, C.H.A. Koster, P. van Bommel (2000), An Evaluation of Linguistically-motivated Indexing Schemes. Proceedings of BCS-IRSG 2000 Colloquium on IR Research, 5th-7th April 2000, Sidney Sussex College, Cambridge, England.

Nuria Bel, Cornelis H.A. Koster and Marta Villegas (2003), Cross-Lingual Text Categorization, Proceedings ECDL 2003, Springer LNCS 2769, pp 126-139.

I.A. Bolshakov (2004), Getting one's first million... collocations, In: CICLing 2004, Springer LNCS 2945, pp 229-242.

P. Bruza and T.W.C. Huibers, (1994), Investigating Aboutness Axioms using Information Fields. Proceedings SIGIR 94, pp. 112-121.

M.F. Caropreso, S. Matwin and F. Sebastiani (2000), A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization, A. G. Chin (Ed.), *Text Databases and Document Management: Theory and Practice*, Idea Group Publishing, Hershey, US, pp. 78-102.

W. B. Cavnar and J. M. Trenkle (1994), N-Gram-Based Text Categorization, Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US, pp 161-175.

W. W. Cohen and Y. Singer (1996), Contextsensitive learning methods for text categorization. In Proceedings of the 19th Annual International ACM Conference on Research and Development in Information Retrieval, pp 307–315.

J.L. Fagan (1988), *Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods*, PhD Thesis, Cornell University.

G. Grefenstette. Light parsing as finite state filtering. In *Workshop on Extended finite state models of language* Budapest, Hungary, August 1996. ECAI'96.

C.H.A. Koster, Affix Grammars for Natural Languages. In: H. Alblas and B. Melichar (Eds.), *Attribute Grammars, applications and systems*. SLNCS 545, Heidelberg, 1991, pp. 469-484.

C.H.A. Koster and M. Seutter (2003), Taming Wild Phrases, Proceedings 25th European Conference on IR Research (ECIR 2003), Springer LNCS 2633, pp 161-176.

C.H.A. Koster and E. Verbruggen (2002), The AGFL Grammar Work Lab, Proceedings of the FREENIX/Usenix conference 2002, Monterey, Ca., pp 13-18.

C.H.A. Koster and P.A. Jones (2004), Best-Only Parsing for Natural Languages. To appear in Proceedings COLING 2004.

W. Kraaij and Renée Pohlmann (1998), Comparing the Effect of Syntactic vs. Statistical Phrase Indexing Strategies for Dutch. Proceedings ECDL 1998, pp 605-617.

Lewis, D.D. (1992), An evaluation of phrasal and clustered representations on a text categorization task. Proceedings ACM SIGIR'92.

G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, *Five Papers on WordNet*. Technical report, Cognitive Science Laboratory, Princeton University, August 1993.

PEKING website: `http://www.cs.kun.nl/peking/`

K. Sparck Jones (1999), The role of NLP in Text Retrieval. In: (Strzalkowski, 1999) pp. 1-24.

T. Strzalkowski (1995), Natural Language Information Retrieval, *Information Processing and Management*, 31 (3), pp. 397-417.

T. Strzalkowski, editor (1999), *Natural Language Information Retrieval*, Kluwer Academic Publishers, ISBN 0-7923-5685-3.

M.A. Alonso and J. Vilares (2002), On the Usefulness of Extracting Syntactic Dependencies for Text Indexing, Springer LNCS 2464, pp. 381-390.

Multiword Units in Syntactic Parsing

Joakim Nivre and Jens Nilsson

School of Mathematics and Systems Engineering
Växjö University
SE-35195 Växjö, Sweden
firstname.lastname@msi.vxu.se

Abstract

The influence of MWU recognition on parsing accuracy is investigated with respect to a deterministic dependency parser based on memory-based learning. The effect of a perfect MWU recognizer is simulated through manual annotation of MWUs in corpus data used for training and testing two different versions of the parser, one which is lexicalized and one which is not. The results show a significant improvement in parsing accuracy not only for MWUs themselves but also with respect to surrounding syntactic structures. The improvement is consistent for both versions of the parser and holds for both labeled and unlabeled accuracy. The greatest improvement corresponds to a 5% error reduction, which is substantial given the relatively low frequency of MWUs in the data. In addition to a quantitative analysis, we also present an analysis of some of the most important error types that are eliminated by the recognition of MWUs.

1. Introduction

The relationship between multiword units (MWUs) and syntactic parsing is a complex one involving several related but distinct questions:

1. Should MWUs be recognized as such and given a special treatment in the syntactic analysis?

2. If the answer is *yes*, is this analysis best performed by the parser itself or by a specialized module applying either before or after the parsing process?

3. If the answer is *no*, can it nevertheless be beneficial to recognize MWUs before or during parsing, even though this is not reflected in the final analysis?

The answer to the first of these questions is largely independent of the parsing problem itself and rather depends on the larger application in which the parsing system is embedded. Moreover, the answer may be different for different classes of MWUs. In information extraction, for example, multiword names (named entities) are highly relevant, whereas compound prepositions are generally not. Questions of this kind are outside the scope of this paper, though. Instead, we will focus on the treatment of MWUs *within* syntactic parsing.

If the first question is dependent on the larger application but largely independent of the specific parsing method used, the inverse probably holds for the second and third question. That is, whether specialized techniques for the analysis of MWUs improves parsing — either of MWUs themselves or of surrounding structures — could be highly dependent on the parsing methodology used. Many parsing frameworks do include a special treatment of multiword units, presumably on the assumption that it improves accuracy (Karlsson et al., 1995; Tapanainen, 1999). However, we have not been able to find any published study where this assumption has been tested in controlled experiments.

Although a conclusive answer to these questions would require us to study a whole range of different parsing techniques, we hope to be able to shed some light on the issues involved by studying two different versions of a memory-based dependency parser, one which is lexicalized and one which is not. More precisely, we have conducted experiments where we simulate a perfect MWU recognizer through manual annotation and study the effect that such a component would have on parsing accuracy, both with respect to the MWUs themselves and with respect to surrounding syntactic structures. It is an open question to what extent the results can be generalized to other parsing methods, but the present study can at least be seen as a first step towards answering the more general questions.

The rest of the paper is structured as follows. First, we describe our general parsing method, which can be characterized as deterministic dependency parsing (section 2). Secondly, we explain how memory-based learning can be used to guide the deterministic parser and we describe the two different versions of the parser used in the experiments (section 3). Next, we report a series of experiments investigating the influence of MWU recognition on parsing accuracy (section 4). Finally, we state our conclusions and give directions for future research (section 5).

2. Deterministic Dependency Parsing

Deterministic dependency parsing has been proposed as a robust and efficient method for syntactic parsing that combines properties of deep and shallow parsing (Yamada and Matsumoto, 2003; Nivre, 2003). Dependency parsing means that the goal of the parsing process is to construct a labeled dependency graph of the kind depicted in Figure 1. Deterministic parsing means that we derive a single analysis for each input string, with no redundancy or backtracking, which makes it possible to parse sentences in linear time (Nivre, 2003).

In formal terms, we define dependency graphs in the following way:

1. Let $R = \{r_1, \ldots, r_m\}$ be the set of dependency types (arc labels).

2. A dependency graph for a word string $W = w_1 \cdots w_n$ is a labeled directed graph $D = (W, A)$, where

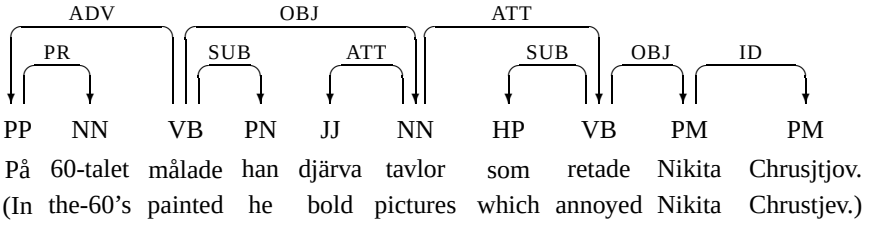


Figure 1: Dependency graph for Swedish sentence

(a) W is the set of nodes, i.e. word tokens in the input string,

(b) A is the arc relation, i.e. a set of labeled directed arcs (w_i, r, w_j) ($w_i, w_j \in W$, $r \in R$),

(c) for every node $w_j \in W$, there is at most one arc $(w_i, r, w_j) \in A$.

3. A graph $D = (W, A)$ is well-formed iff it is acyclic, projective and connected.

The parsing algorithm used here was first defined for unlabeled dependency parsing (Nivre, 2003) and subsequently extended to labeled graphs (Nivre et al., 2004). Parser configurations are represented by triples $\langle S, I, A\rangle$, where S is the stack (represented as a list), I is the list of (remaining) input tokens, and A is the (current) arc relation for the dependency graph. (Since in a dependency graph the set of nodes is given by the input tokens, only the arcs need to be represented explicitly.) Given an input string W, the parser is initialized to $\langle \mathbf{nil}, W, \emptyset\rangle$ and terminates when it reaches a configuration $\langle S, \mathbf{nil}, A\rangle$ (for any list S and set of arcs A). Given an arbitrary configuration of the parser, there are four possible transitions to the next configuration:

1. **Left-Arc:** In a configuration $\langle t|S,n|I,A\rangle$, if there is no arc $(w, r, t) \in A$, extend A with (n, r', t) and pop the stack, giving the configuration $\langle S,n|I,A\cup\{(n, r', t)\}\rangle$.

2. **Right-Arc:** In a configuration $\langle t|S,n|I,A\rangle$, if there is no arc $(w, r, n) \in A$, extend A with (t, r', n) and push the token n onto the stack, giving the configuration $\langle n|t|S,I,A \cup \{(t, r', n)\}\rangle$.

3. **Reduce:** In a configuration $\langle t|S,I,A\rangle$, if there is an arc $(w, r, t) \in A$, pop the stack, giving the configuration $\langle S,I,A\rangle$.

4. **Shift:** In a configuration $\langle S,n|I,A\rangle$, push n onto the stack, giving the configuration $\langle n|S,I,A\rangle$.

After initialization, the parser is guaranteed to terminate after at most $2n$ transitions, given an input string of length n (Nivre, 2003). Moreover, the parser always constructs a dependency graph that is acyclic and projective. This means that the dependency graph given at termination is well-formed if and only if it is connected (Nivre, 2003).

The transition system defined above is nondeterministic in itself, since several transitions can often be applied in a given configuration. To construct deterministic parsers based on this system, we use classifiers trained on treebank data in order to predict the next transition (and dependency type) given the current configuration of the parser. In the experiments reported here, we use memory-based learning to train these classifiers.

3. Memory-Based Learning

Memory-based learning and problem solving is based on two fundamental principles: learning is the simple storage of experiences in memory, and solving a new problem is achieved by reusing solutions from similar previously solved problems (Daelemans, 1999). It is inspired by the nearest neighbor approach in statistical pattern recognition and artificial intelligence (Fix and Hodges, 1952), as well as the analogical modeling approach in linguistics (Skousen, 1989; Skousen, 1992). In machine learning terms, it can be characterized as a lazy learning method, since it defers processing of input until needed and processes input by combining stored data (Aha, 1997).

Memory-based learning has been successfully applied to a number of problems in natural language processing, such as grapheme-to-phoneme conversion, part-of-speech tagging, prepositional-phrase attachment, and base noun phrase chunking (Daelemans et al., 2002). It has also been used to guide deterministic parsers (Veenstra and Daelemans, 2000; Nivre et al., 2004).

For the experiments reported in this paper, we have used the software package TiMBL (Tilburg Memory Based Learner), which provides a variety of metrics, algorithms, and extra functions on top of the classical k nearest neighbor classification kernel, such as value distance metrics and distance weighted class voting (Daelemans et al., 2003).

The function we want to approximate is a mapping f from configurations to parser actions, where each action consists of a transition and (except for **Shift** and **Reduce**) a dependency type:

$$f : \mathit{Config} \rightarrow \{\mathbf{LA}, \mathbf{RA}, \mathbf{RE}, \mathbf{SH}\} \times (R \cup \{\mathbf{nil}\})$$

Here Config is the set of all possible parser configurations and R is the set of dependency types. However, in order to make the problem tractable, we try to learn a function $\hat{f}$ whose domain is a finite space of parser *states*, which are abstractions over configurations. For this purpose we define a number of features that can be used to define different models of parser state. The features used in this study are listed in Table 1.

The first five features (TOP–TOP.RIGHT) deal with properties of the token on top of the stack. In addition to the

Feature	Description
TOP	The token on top of the stack
TOP.POS	The part-of-speech of TOP
TOP.DEP	The dependency type of TOP (if any)
TOP.LEFT	The dependency type of TOP's leftmost dependent (if any)
TOP.RIGHT	The dependency type of TOP's rightmost dependent (if any)
NEXT	The next input token
NEXT.POS	The part-of-speech of NEXT
NEXT.LEFT	The dependency type of NEXT's leftmost dependent (if any)
LOOK.POS	The part-of-speech of the next plus one input token

Table 1: Parser state features

word form itself (TOP), we consider its part-of-speech (as assigned by an automatic part-of-speech tagger in a preprocessing phase), the dependency type by which it is related to its head (which may or may not be available in a given configuration depending on whether the head is to the left or to the right of the token in question), and the dependency types by which it is related to its leftmost and rightmost dependent, respectively (where the current rightmost dependent may or may not be the rightmost dependent in the complete dependency tree).

The following three features (NEXT–NEXT.LEFT) refer to properties of the next input token. In this case, there are no features corresponding to TOP.DEP and TOP.RIGHT, since the relevant dependencies can never be present at decision time. The final feature (LOOK) is a simple lookahead, using the part-of-speech of the next plus one input token.

In the experiments reported below, we have used two different parser state models, one called the *lexical* model, which includes all nine features, and one called the *non-lexical* model, where the two lexical features TOP and NEXT are omitted. The learning algorithm used is the IB1 algorithm (Aha et al., 1991) with k = 5, i.e. classification based on 5 nearest neighbors.[1] Distances are measured using the modified value difference metric (MVDM) (Stanfill and Waltz, 1986; Cost and Salzberg, 1993), and classification is based on distance weighted class voting with inverse distance weighting (Dudani, 1976). For more information about the different parameters and settings, see Daelemans et al. (2003).

4. Experiments

The data used for the experiments come from a manually annotated corpus of written Swedish, created at Lund University in the 1970's and consisting mainly of informative texts from official sources (Einarsson, 1976). Although the original annotation scheme is an eclectic combination of constituent structure, dependency structure, and topological fields (Teleman, 1974), it has proven possible to convert the annotated sentences to dependency trees with fairly high accuracy. What makes this corpus especially suitable for this study, is the fact that several classes of MWUs have been annotated in such a way that they can be identified automatically. These expressions can be divided into three broad classes, exemplified below:

- Multiword names (MN):
  Torsten Nilsson (person)
  Västra Frölunda (place)
  Södra Kyrkogatan (street)
- Numerical expressions (NE):
  3 - 4
  6 X 6
  1967 / 68
- Compound function words, which can be subdivided into five categories:
  1. Adverbs (AB):
     så småningom (eventually)
     i allmänhet (in general)
     var som helst (anywhere)
  2. Prepositions (PP):
     på grund av (because of)
     med hänsyn till (with respect to)
     i samband med (in connection with)
  3. Subordinating conjunctions (SN):
     efter det att (after)
     även om (even if)
     trots att (despite the fact that)
  4. Determiners (DT):
     den här (this)
     en och samma (one and the same)
     en del (some)
  5. Pronouns (PN):
     den här (this)
     var och en (everyone)
     en del (some)

Table 2 gives frequency counts for different categories of MWUs in the corpus and compares them to the overall statistics for tokens and sentences. We see that compound function words, taken together, is the largest class, followed by multiword names, whereas numerical expressions are rare in the corpus. The overall frequency of the annotated MWUs can be appreciated by noting that, on average, there

[1] In TiMBL, the value of k in fact refers to k nearest distances rather than k nearest neighbors, which means that, even with $k = 1$, the nearest neighbor set can contain several instances that are equally distant to the test instance. This is different from the original IB1 algorithm (Aha et al., 1991).

Category	Example	Frequency
MN	Torsten Nilsson	427
NE	3 - 4	10
AB	så småningom (eventually)	826
PP	på grund av (because of)	256
SN	efter det att (after)	167
DT	den här (this)	94
PN	den här (this)	30
MWUs		1810
Sentences		6316
Tokens		97623

Table 2: Statistics of the data set

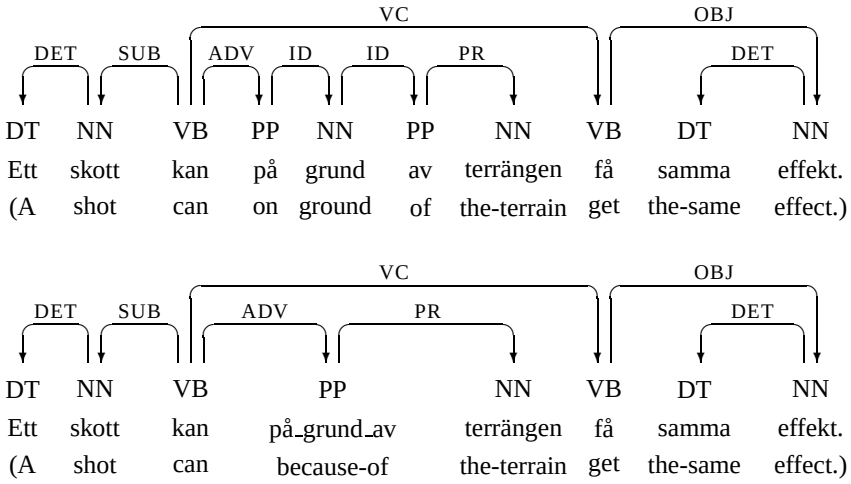


Figure 2: Treebank representations: Baseline (top) and MWU (bottom)

are 3 MWUs in 10 sentences and 2 MWUs in 100 words. It should be emphasized, however, that this only takes into account the MWUs that could be identified through the original corpus annotation. However, the overall frequency figures accord well with results from automatic acquisition of MWUs with reported extraction rates of 1–3 MWUs per 100 words (Dias et al., 1999).

For the experiments we have created two different versions of the treebank. In the baseline version, each MWU is represented as a sequence of tokens, each with its own part-of-speech, combined into a left-headed dependency chain with arcs labeled with the special symbol ID. In the MWU version, each MWU is represented by a single token, which is formed by joining the constituent tokens with underscores (e.g. *på grund av* → *på_grund_av*) and assigned a part-of-speech appropriate for the entire MWU.

By way of illustration, Figure 2 shows one and the same sentence in both representations. The experiments consist in training and testing the memory-based dependency parser on the two different versions of the treebank, measuring the parsing accuracy both for MWUs themselves (under the baseline condition) and for all other syntactic structures (both conditions). All experiments have been carried out with both the lexical and the non-lexical versions of the parser.

For other experiments, the treebank has been divided into three non-overlapping data sets: 80% for training 10% for development/validation, and 10% for final testing (random samples). The results presented in this study are all from the validation set, i.e. all parsers have been trained on the training set and evaluated on the validation set. (The final test set has not been used at all in the experiments reported in this paper.)

Parsing accuracy is measured in the experiments by three different metrics:

- **Attachment score (AS):** The proportion of tokens (excluding punctuation) that are assigned the correct head (or no head if the token is a root) (Eisner, 1996; Collins et al., 1999).
- **Labeled attachment score (LAS):** The proportion of tokens (excluding punctuation) that are assigned both the correct head and the correct dependency type (or no head if the token is a root).
- **MWU accuracy:** Labeled precision, recall and $F_{\beta=1}$ score for the special ID relation (only applicable under the baseline condition).

Table 3 shows the accuracy with which the parsers can learn to recognize MWUs by combining their component

Model	Prec	Rec	$F_{\beta=1}$
Non-lexical	55.6	27.4	36.7
Lexical	75.7	67.1	71.1

Table 3: Parsing accuracy for MWUs only

word tokens into left-headed dependency chains labeled with the dedicated ID relation. We see that none of the parsers trained on the baseline data achieve a very high accuracy. Not surprisingly, the lexicalized parser performs best, with 75.7% precision and 67.1% recall for the ID relation, while the non-lexicalized parser only reaches 55.6% precision and 27.4% recall. Although these results are perhaps not very interesting in themselves, we can at least take the performance of the lexicalized parser, corresponding to an $F_{\beta=1}$ score of 71.1%, as a baseline for further experiments with fully automatic MWU recognition. If the overall accuracy is going to improve, a dedicated MWU recognizer will at least have to do better than the parser on its own.

Table 4 compares the accuracy of the parsers when trained and tested on the two versions of the data set. We see that both parsers perform consistently better with the MWU representation than with the baseline representation, even if errors in the MWUs themselves are not counted. With respect to the basic attachment score, there is an improvement from 83.0% to 83.5% for the non-lexical parser, and from 84.7% to 85.3% for the lexical one. With respect to labeled attachment, the improvement is even greater. The non-lexical parser goes from 76.1% to 77.4%, and the lexical parser from 80.7% to 81.6%. All differences are significant beyond the 0.0001 level (McNemar's test).

The experimental results show unequivocally that the recognition of MWUs improves parsing accuracy not only with respect to the MWUs themselves, but also with respect to surrounding syntactic structures. The differences are quantitatively fairly small, and the largest gain only corresponds to a 5% error reduction, but we have to keep in mind that MWUs are relatively rare themselves. We noted earlier that there are about 2 MWUs per 100 words. So, if parsing accuracy improves by 1% then, roughly speaking, one MWU in two makes a difference. It is therefore not surprising that the differences are highly statistically significant, even though they are numerically small compared to the total number of parsing errors.

On the other hand, we must also remember that the MWU representation used in the experiments is a simulation of 100% accurate MWU recognition, which is of course impossible to achieve in practice. It remains to be seen how much of the theoretically possible improvement can be realized when using automatic methods for MWU recognition.

If we look in detail at the kind of parsing errors that are eliminated under the MWU condition, they can be divided into two broad classes:

- Errors that occur in the vicinity of a MWU and where the baseline parsers are misled either by an incorrect analysis of the MWU or by the fact that the MWU is not recognized as a lexical unit. We will call these MIC errors (for "MWU in context").

- Errors that do not occur in the vicinity of a MWU but which are due to "noise" in the training data resulting from the failure to recognize MWUs as lexical units. We will call these MOC errors (for "MWU out of context").

MIC errors can be further subdivided into several types, and although an exhaustive analysis of these errors is beyond the scope of this paper, we will discuss two of the most important types here. The first type occurs when the internal structure of the MWU is anomalous given its syntactic function as a unit. A good example is found in Figure 3, involving the MWU *vad beträffar* (as regards), where the second word token is a finite verb. Thus, the lexical baseline parser analyzes this as a unit with clause structure, treating the *wh*-pronoun *vad* (what) as the subject of the verb *beträffar* (regards), and the following conjoined noun phrase *jordbruk och industri* (agriculture and industry) as the object of this verb. By contrast, the lexical MWU parser recognizes *vad beträffar* as a compound preposition, which yields a correct analysis of the entire structure.

The second type of error occurs when the internal structure of the MWU is compatible with its syntactic function, but where the fact that it is not recognized as a lexical unit results in an incorrect analysis of the surrounding structure, typically in the form of an attachment error. Figure 4 gives an example involving the compound adverb *i regel* (as a rule), which is analyzed as an ordinary prepositional phrase by the lexical baseline parser. Now, this does not lead to an incorrect analysis of its syntactic function, since the adverbial function is perfectly compatible with a prepositional phrase analysis. However, it leads to an incorrect attachment decision in that the following agent phrase *av kommuner* (by communes) is attached as an attribute to the noun *regel* (rule), instead of being treated as an adverbial to the passive verb *drivs* (is run). Again, the MWU parser avoids this problem, because the unit *i regel* is not only assigned an adverbial function but is also treated as a compound adverb.

In the foregoing example, the error consists in making an incorrect attachment *to* the MWU, but it is also common for the baseline parsers to make incorrect attachments *of* MWUs, where the corresponding MWU parsers make correct attachments. A rather amusing example is the following:

Condition	Non-lexical		Lexical	
	AS	LAS	AS	LAS
Baseline	83.0	76.1	84.7	80.7
MWU	83.5	77.4	85.3	81.6

Table 4: Parsing accuracy excluding MWUs

Vi skulle ha stoppat dem i kassetter i stället för att slicka igen kuvert.

Baseline: We should have put them into cassettes in the rack in order to seal envelopes.

MWU: We should have put them into cassettes instead of sealing envelopes.

While the lexical MWU parser correctly analyzes *i stället för* (instead of) as a compound preposition introducing an adverbial modifier, the baseline parser treats *i stället* (in the place) as a prepositional phrase, attaching it as a post-modifier to the noun *kassetter* (cassettes). However, since the word form *stället* is ambiguous and can also mean "the rack", and since *för att* can be analyzed as a compound subordinating conjunction (in order to), it is actually possible to derive a coherent (but contextually incorrect) analysis of the sentence where *i stället* is indeed a prepositional phrase, possibly acting as a nominal post-modifier.

MOC errors are harder to categorize than MIC errors since they are more indirectly related to the analysis of MWUs as such. But we can illustrate the phenomenon with a case that is fairly frequent in the Swedish data, namely the two-word sequence *så att*. This can either be a multiword subordinating conjunction with a consecutive meaning (corresponding to *so that* in English), or the adverb *så* (so, like that) followed by either the subordinating conjunction *att* (that) or the infinitive marker *att* (to). Now, when trained on a corpus where these two cases are only distinguished through their syntactic analysis and not as one versus two lexical units, the parser will have to learn to disambiguate them syntactically. This will typically result in errors going in both directions, i.e. not only MWUs being analyzed as syntactic combinations but also syntactic combinations being analyzed as MWUs. And whereas the former type of error can be eliminated by recognizing MWUs prior to parsing new input, the latter type will remain unless the same MWU recognition is also applied to the training data for the parser.

Finally, it may be worth noting that the recognition of MWUs also improves the robustness of the parsers. Thus, while the lexical baseline parser fails to produce a well-formed dependency graph (i.e. a complete projective tree) for 13.3% of the sentences in the test data set, the corresponding figure for the lexical MWU parser is only 10.8%. (The corresponding figures for the non-lexical parser are 8.8% and 7.8%.) An inspection of the problematic sentences reveals that accuracy and robustness go hand in hand in this case. We have seen examples above where the baseline parsers make the wrong attachment because they fail to recognize MWUs properly. In the case of incomplete sentences, the consequence is instead that the parsers make *no* attachment. But the cause of the problem is the same in both cases.

To sum up, we have seen that the recognition of MWUs can improve the quality of data-driven parsing in several different ways. First of all, it may improve the consistency of the training data, by eliminating word sequences that can be interpreted both as MWUs and as syntactic combinations, which leads to an improvement in overall parsing accuracy. Secondly, it can eliminate parsing errors both within MWUs and in their relation to surrounding syntactic structures. Finally, the gain in accuracy may also lead to an improvement in robustness.

5. Conclusion

In this paper, we have investigated the influence of MWU recognition on parsing accuracy for a deterministic dependency parser based on memory-based learning. We have simulated the effect of a perfect MWU recognizer through manual annotation of MWUs, in particular multiword names and compound function words, in corpus data used for training and testing two different versions of the parser, one lexicalized and one non-lexicalized.

The results show a significant improvement in parsing accuracy not only for MWUs themselves, but also with respect to surrounding syntactic structures. The improvement holds for both labeled and unlabeled attachment accuracy and is consistent for both versions of the parser. The greatest improvement (labeled attachment for the non-lexical parser) corresponds to a 5% error reduction, which is substantial given the relatively low frequency of MWUs in the data. In addition to the quantitative analysis, we have also presented an analysis of some of the most important error types that are eliminated by the recognition of MWUs, making a distinction between errors that occur in the context of MWUs (MIC errors) and errors that occur in other contexts but are due to the presence of unanalyzed MWUs in the training data (MOC errors).

The results presented in this study can be taken as an indication of the maximum gain in parsing accuracy that can be achieved by using MWU recognition prior to training and parsing, at least when restricted to the MWU categories represented in our data. An important topic for future research is to see how much of this potential can be realized in practice, when relying on automatic recognition of MWUs rather than manually annotated corpus data. However, it should also be remembered that since the original corpus annotation in our case was done for a different purpose than facilitating syntactic parsing, we do not know to what extent the recognized categories of MWUs are the most suitable to improve parsing performance. This may mean that the potential for improvement is actually greater than indicated by our experimental results.

Finally, it should be pointed out again that we have really only considered a special case of the general questions formulated in the introduction. What we have shown is that for a particular kind of parser, a deterministic dependency parser guided by memory-based learning, there is a small but significant gain in accuracy that results from recognizing MWUs prior to training and parsing. It is an open question how far these results can be generalized to other parsing methods, in particular methods that are grammar-based rather than data-driven.

6. Acknowledgements

The work presented in this paper was supported by a grant from the Swedish Research Council (621-2002-4207). The memory-based classifiers used in the experiments were constructed using the Tilburg Memory-Based Learner (TiMBL) (Daelemans et al., 2003).

7. References

Aha, D. (ed.), 1997. *Lazy Learning*. Kluwer.

Aha, D. W., D. Kibler, and M. Albert, 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

Collins, Michael, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann, 1999. A Statistical Parser of Czech. In *Proceedings of 37th ACL Conference*. University of Maryland, College Park, USA.

Cost, S. and S. Salzberg, 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

Daelemans, Walter, 1999. Memory-based language processing. introduction to the special issue. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:287–292.

Daelemans, Walter, Antal van den Bosch, and Jakub Zavrel, 2002. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11–43.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch, 2003. Timbl: Tilburg memory based learner, version 5.0, reference guide. Technical Report ILK 03-10, Tilburg University, ILK.

Dias, Gaël, Sylvie Guilloré, and Gabriel Lopes, 1999. Language independent automatic acquisition of rigid multiword units from unrestricted text corpora. In *Actes Traitement Automatique des Langues Naturelles*.

Dudani, S. A., 1976. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6:325–327.

Einarsson, Jan, 1976. Talbankens skriftspråkskonkordans. Lund University.

Eisner, Jason M., 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*. Copenhagen.

Fix, E. and J. Hodges, 1952. Discriminatory analysis: Nonparametric discrimination: Consistency properties. Technical Report 11, USAF School of Aviation Medicine, Randolph Field, Texas.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, and Arto Anttila (eds.), 1995. *Constraint Grammar: A language-independent system for parsing unrestricted text*. Number Volume 4 in Natural Language Processing. Mouton de Gruyter.

Nivre, Joakim, 2003. An efficient algorithm for projective dependency parsing. In Gertjan van Noord (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*.

Nivre, Joakim, Johan Hall, and Jens Nilsson, 2004. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*.

Skousen, Royal, 1989. *Analogical Modeling of Language*. Kluwer.

Skousen, Royal, 1992. *Analogy and Structure*. Kluwer.

Stanfill, C. and D. Waltz, 1986. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228.

Tapanainen, Pasi, 1999. *Parsing in Two Frameworks: Finite-State and Functional Dependency Grammar*. Ph.D. thesis, University of Helsinki.

Teleman, Ulf, 1974. *Manual för grammatisk beskrivning av talad och skriven svenska*. Studentlitteratur.

Veenstra, Jorn and Walter Daelemans, 2000. A memory-based alternative for connectionist shift-reduce parsing. Technical Report ILK-0012, University of Tilburg.

Yamada, Hiroyasu and Yuji Matsumoto, 2003. Statistical dependency analysis with support vector machines. In Gertjan van Noord (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*.

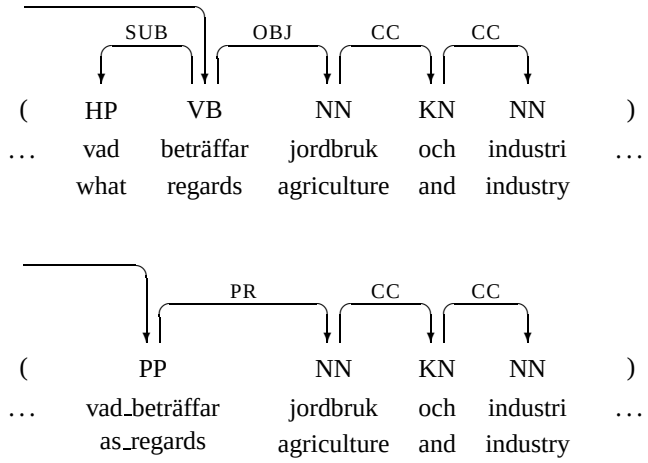


Figure 3: Syntactically anomalous MWU: Baseline (top) and MWU (bottom)

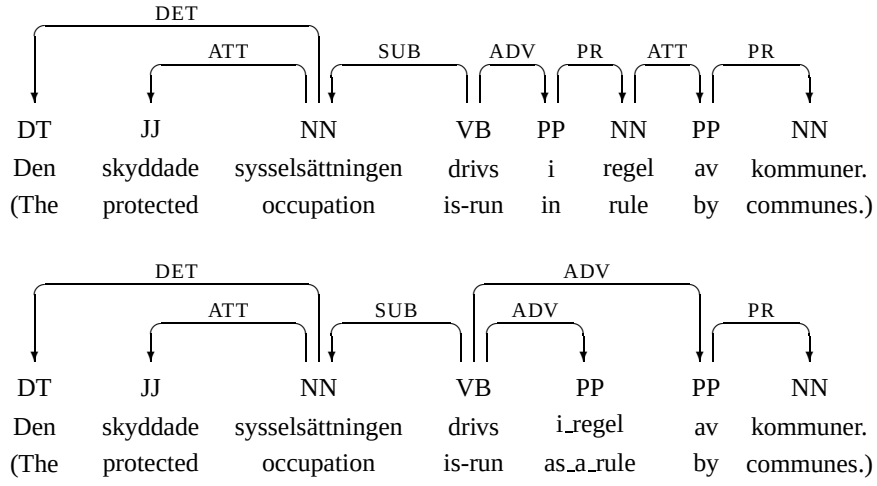


Figure 4: Attachment error: Baseline (top) and MWU (bottom)

Use of Noun Phrases in Interactive Search Refinement

Olga Vechtomova
Department of Management Sciences
University of Waterloo
200 University Avenue West, Waterloo, Canada
ovechtom@engmail.uwaterloo.ca

Murat Karamuftuoglu
Department of Computer Engineering
Bilkent University
06800 Bilkent Ankara, Turkey
hmk@cs.bilkent.edu.tr

Abstract

The paper presents an approach to interactively refining user search formulations and its evaluation in the new High Accuracy Retrieval from Documents (HARD) track of TREC-12. The method consists of showing to the user a list of noun phrases, extracted from the initial document set, and then expanding the query with the terms taken from the phrases selected by the user. The results show that the method yielded significant gains in retrieval performance. The paper also discusses post-TREC experiments conducted to explore the use of Pointwise Mutual Information measure in selecting multiword units for query expansion and the use of n-grams in the search process.

1 Introduction

Query expansion following relevance feedback is a well-established technique in information retrieval, which aims at improving user search performance. It combines user and system effort towards selecting and adding extra terms to the original query. The traditional model of query expansion following relevance feedback is as follows: the user reads a representation of a retrieved document, typically its full-text or abstract, and provides the system with a binary relevance judgement. After that the system extracts query expansion terms from the document, which are added to the query either manually by the searcher – interactive query expansion, or automatically – automatic query expansion. Intuitively interactive query expansion should produce better results than automatic, however this is not consistently the case (Beaulieu 1997, Koenemann and Belkin 1996, Ruthven 2003).

In this paper we present a new approach to interactive query expansion, which we developed and tested within the framework of the High Accuracy Retrieval from Documents (HARD) track of TREC (Text Retrieval Conference).

1.1 HARD track

The Text REtrieval Conference (TREC), co-sponsored by the National Institute of Standards and Technology and U.S. Department of Defense, was started in 1992 to support research into large-scale evaluation of text retrieval methodologies.

The main goal of the new HARD track in TREC-12 is to explore what techniques could be used to improve search results by using two types of information:

1. Extra-linguistic contextual information about the user and the information need, which was provided by track organisers in the form of metadata. It specifies the following:

- *Genre* – the type of documents that the searcher is looking for. It has the following values:
  - Overview (general news related to the topic);
  - Reaction (news commentary on the topic);
  - I-Reaction (as above, but about non-US commentary)
  - Any.
- *Purpose* of the user's search, which has one of the following values:
  - Background (the searcher is interested in the background information for the topic);
  - Details (the searcher is interested in the details of the topic);
  - Answer (the searcher wants to know the answer to a specific question);
  - Any.
- *Familiarity* of the user with the topic on a five-point scale.
- *Granularity* – the amount of text the user is expecting in response to the query. It has the following values: Document, Passage, Sentence, Phrase, Any.
- *Related text* – sample relevant text found by the users from any source, except the evaluation corpus.

2. Relevance feedback given by the user in response to topic clarification questions. This information was elicited by each site by means of a (manually or automatically) composed set of clarification forms per topic. The forms are filled in by the annotators (users), and provide additional search criteria.

In more detail the HARD track evaluation scenario consists of the following steps:

1) The track organisers invite annotators, each of whom formulates one or more topics. An example of a typical HARD topic is given below:

Title: Red Cross activities

Description: What has been the Red Cross's international role in the last year?

Narrative: Articles concerning the Red Cross's activities around the globe are on topic. Has the RC's role changed? Information restricted to international relief efforts that do not include the RC are off-topic.

Purpose: Details

Genre: Overview

Granularity: Sentence

Familiarity: 2

2) Participants receive Title, Description and Narrative sections of the topics, and use any information from them to produce one or more baseline runs.

3) Participants produce zero or more clarification forms with the purpose of obtaining feedback from the annotators. Only two forms were guaranteed to be filled out and returned. According to the HARD track specifications, a clarification form for each topic must fit into a screen with 1152 x 900 pixels resolution, and the user may spend no more than 3 minutes filling out each form.

4) All clarification forms from different sites for a topic are filled out by the annotator, who has composed that topic.

5) Participants receive the topic metadata and the annotators' responses to clarification forms, and use any data from them to produce one or more final runs.

6) Two runs per site (baseline and final) are judged by the annotators. Top 75 documents, retrieved for each topic in each of these runs, are assigned binary relevance judgement by the annotator – author of the topic.

7) The annotators' relevance judgements are then used to calculate the performance metrics (see section 3).

The evaluation corpus used in the HARD track consists of 372,219 documents, and includes three newswire corpora (New York Times, Associated Press Worldstream and Xinghua English) and two governmental corpora (The Congressional Record and Federal Register). The overall size of the corpus is 1.7Gb.

The primary goal of our participation in the track was to investigate how to achieve high retrieval accuracy through relevance feedback. The secondary goal was to study ways of reducing the amount of time and effort the user spends on making a correct relevance judgement.

Traditionally in bibliographical and library IR systems the hitlist of retrieved documents is represented in the form of the titles and/or the first few sentences of each document. Based on this information the user has to make initial implicit relevance judgements: whether to refer to the full text document or not. Explicit relevance feedback is typically requested by IR systems after the user has seen the full-text document, an example of such IR system is Okapi (Robertson et al. 2000, Beaulieu 1997). Reference to full text documents is obviously time-consuming, therefore it is important to represent documents in the hitlist in such a form, that would enable the users to reliably judge their relevance without referring to the full text. Arguably, the title and the first few sentences of the document are frequently not sufficient to make correct relevance judgement. Query-biased summaries, usually constructed through the extraction of sentences that contain higher proportion of query terms than the rest of the text – may contain more relevance clues than generic document representations. Tombros and Sanderson (1998) compared query-biased summaries with the titles plus the first few sentences of the documents by how many times the users have to request full-text documents to verify their relevance/non-relevance. They discovered that subjects using query-biased summaries refer to the full text of only 1.32% documents, while subjects using titles and first few sentences refer to 23.7% of documents. This suggests that query-biased representations are likely to contain more relevance clues than generic document representations.

We have experimented with a similar approach in HARD track. Given the restrictions on the available space for relevance feedback, we created micro-summaries that consisted of single sentences for each of the top ranked documents in the baseline run. The sentences were selected according to concentration of the content-bearing and query-related words in them. The users were asked to select those sentences that might indicate relevant documents. Contrary to our preliminary experiments, the official HARD track runs for this method was not successful (Vechtomova 2004). In this paper we, therefore, describe the more successful method of directly eliciting query expansion terms from the users and evaluation of its effectiveness in HARD track of TREC 2003.

The method extracts noun phrases from top-ranked documents retrieved in the baseline run and asks the user to select those, which might be useful in retrieving relevant documents. The selected phrases are then used in constructing an expanded query, which retrieves a new set of documents. This approach aims to minimise the amount of text the user has to read in relevance feedback, and to focus the user's attention on the key information clues from the documents.

The remainder of this paper is organised as follows: section 2 presents the query expansion method we developed, section 3 discusses its evaluation, sections 4 and 5 describe post-TREC experiments we have conducted with phrases. Section 6 concludes the paper and outlines future research directions.

2 Query Expansion Method

The user feedback mechanism that we evaluated consists of automatically selecting noun phrases from the top-ranked documents retrieved in the baseline run, and asking the users to select all phrases that contain possibly useful query expansion terms.

The research question explored here is whether noun phrases provide sufficient context for the user to select potentially useful terms for query expansion.

An important question in query expansion is which part of the document should be used in extracting expansion terms/phrases. Two common approaches in IR are: (1) to extract candidate terms from the entire document; (2) to extract them only from the best matching passages. The rationale for the second approach is that documents may be about multiple topics, not all of which are relevant to the user's query, therefore we would reduce the amount of noise by extracting terms/phrases only from those parts of the documents, which are likely

to be related to the user's query.

We developed a method of selecting sentences in the documents, which are (1) most likely to be related to the query, and (2) have high information density. The best *n* sentences are then used for extracting noun phrases. In more detail the sentence selection algorithm is outlined below.

In all our experiments we used an experimental IR system *Okapi* (Robertson et al. 2000), and its best-match search function *BM25*.

2.1 Sentence selection

The sentence selection algorithm consists of the following steps:

We take N top-ranked documents, retrieved in response to query terms from the topic title. The full-text of each of the documents is then split into sentences. For every sentence that contains one or more query terms, i.e. any term from the title field of the topic statement, two scores are calculated: S1 and S2.

Sentence selection score 1 (S1) is the sum of *idf* of all query terms present in the sentence.

$$S1 = \sum idf_q \qquad (1)$$

Sentence selection score 2 (S2):

$$S2 = \frac{\sum W_i}{f_s} \qquad (2)$$

Where: W_i – Weight of the term *i*, see (3);
f_s – length normalisation factor for sentence *s*, see (4).

The weight of each term in the sentence, except stopwords, is calculated as follows:

$$W_i = idf_i(0.5 + (0.5 * \frac{tf_i}{t\max})) \qquad (3)$$

Where: idf_i – inverse document frequency of term *i* in the corpus; tf_i – frequency of term *i* in the document; *tmax* – *tf* of the term with the highest frequency in the document.

To normalise the length of the sentence we introduced the sentence length normalisation factor *f*:

$$f_s = \frac{s\max}{slen_s} \qquad (4)$$

Where: *smax* – the length of the longest sentence in the document, measured as the number of non-stopwords it contains; *slen* – the length of the current sentence.

All sentences in the document were ranked by S1 as the primary score and S2 as the secondary score. Thus, we first select the sentences that contain more query terms, and therefore are more likely to be related to the user's query, and secondarily, from this pool of sentences select the one which is more content-bearing, i.e. containing a higher proportion of terms with high *tf***idf* weights.

2.2 Noun phrase selection

We take top 25 documents from the baseline run, and select 2 sentences per document using the algorithm described above. We have not experimented with alternative values for these two parameters.

We then apply Brill's rule-based tagger (Brill 1995) and BaseNP noun phrase chunker (Ramshaw and Marcus 1995) to extract noun phrases from these sentences.

The phrases are then parsed in Okapi to obtain their term weights, removing all stopwords and phrases consisting entirely of the original query terms. The remaining phrases are ranked by the sum of weights of their constituent terms. Top 78 phrases are then included in the clarification form for the user to select. This is the maximum number of phrases that could fit into the clarification form.

All user-selected phrases were split into single terms, which were then used to expand the original user query. The expanded query was then searched against the HARD track database using Okapi BM25 search function. The official TREC evaluation results are discussed in section 3.

Following TREC 2003 we have also experimented with:

1) the use of phrases in searching instead of single terms;
2) the use of an association measure (pointwise mutual information) in selecting noun phrases for query expansion.

These experiments and their results are discussed in sections 4 and 5.

3 Evaluation

The runs submitted to the HARD track were evaluated in three different ways. The first two evaluations are done at the document level only, whereas the last one takes into account the granularity metadata.

1. SOFT-DOC – document-level evaluation, where only the traditional TREC topic formulations (title, description, narrative) are used as relevance criteria.
2. HARD-DOC – the same as the above, plus 'purpose', 'genre' and 'familiarity' metadata are used as additional relevance criteria.
3. HARD-PSG – passage-level evaluation, which in addition to all criteria in HARD-DOC also requires that retrieved items satisfy the granularity metadata (Allan 2004).

Document-level evaluation was done by the traditional IR metrics of mean average precision and precision at various document cutoff points. In this paper we focus on document-level evaluation. Passage-level evaluation is discussed elsewhere (Vechtomova et al. 2004).

3.1 Document-level evaluation

For all of our runs we used Okapi BSS (Basic Search System). For the baseline run we used keywords from the title field only, as these proved to be most effective in our preliminary experiments. Topic titles were parsed in Okapi, weighted and searched using BM25 function against the HARD track corpus.

Document-level results of the submitted runs are given in table 1. UWAThard1 is the baseline run using original query terms from the topic titles. UWAThard3 is an experimental run using the query expansion method described earlier. Query expansion resulted in 18% increase in average precision (SOFT-DOC evaluation) and 26.4% increase in average precision (HARD-DOC evaluation). Both improvements are statistically significant (using t-test at .05 significance level). On average 19 phrases were selected by users per topic.

Run	SOFT-DOC Evaluation		HARD-DOC evaluation	
	P@ 10	AveP	P@ 10	AveP
UWAThard1 (baseline run)	0.4875	0.3134	0.3875	0.2638
UWAThard3 (experimental run)	0.5958	0.3719	0.4854	0.3335

Table 1: Document-level evaluation results

In total 88 runs were submitted by participants to the HARD track. All our submitted runs are above the median in all evaluation measures shown in table 1. The only participating site, whose expansion runs performed better than our UWAThard3 run, was the Queen's college group (Kwok et al. 2004). Their best baseline system achieved 32.7% AveP (HARD-DOC) and their best result after clarification forms was 36%, which gives 10% increase over the baseline. We have achieved 26% improvement over the baseline (HARD-DOC), which is the highest increase over baseline among the top 50% highest-scoring baseline runs.

3.2. Analysis of performance by topic

We have conducted a topic-by-topic analysis of its performance in comparison with the baseline. Figure 1 shows the average precision (HARD-DOC) of these two runs by topic. It is not surprising, that performance of query expansion following blind feedback tends to depend on performance of the original query. The fewer relevant documents are retrieved at the top of the ranked list by the original query, the fewer good candidate query expansion terms are extracted, and hence the lower is the performance of the expanded run. This tendency is evident from Figure 1.

We have analysed two groups of topics: (1) topics, which yielded significantly worse results in runs with the expanded query (UWAThard3) than runs with the original query terms (baseline); and (2) topics, which had low performance both with the original and the expanded queries. Some examples of topic titles in the first group are: "Corporate mergers" (topic 222), "Sports scandals" (223), "Oscars" (53) and "IPO activity" (196). One factor that all of these topics have in common is that query expansion phrases selected by the users from the

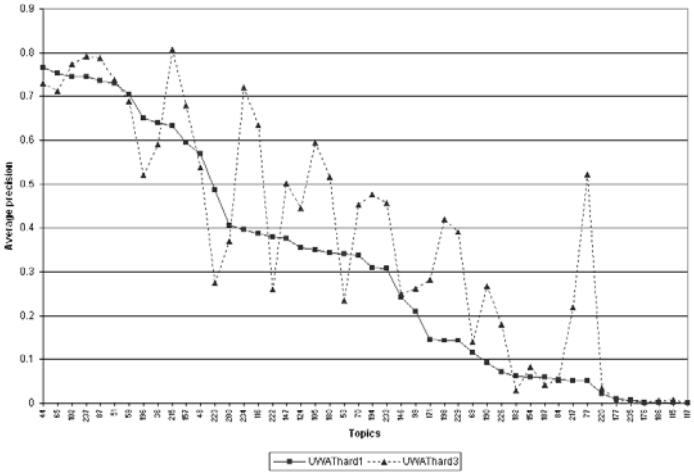


Figure 1: Results by topic of the baseline (UWAThard1) and the query expansion run (UWAThard3)

candidate phrases shown to them contain a large number of proper names.

Generally, proper names are considered to be good candidates for query expansion, as they usually have relatively low collection frequency. However, in our current model, we break user-selected multi-term phrases into their constituent terms and use them in the search process. For example, a proper name "Dan Leonard" selected by users for query expansion in topic 223 ("Sports scandals") was decomposed into single terms, each of which could match references to unrelated individuals. This results in many false matches. The situation is also aggravated in many cases by high *idf* values of some of the proper name components, which dominate the search results.

Examples of topic titles in the second group are: "National leadership transitions" (187), "School development" (182), "Virtual defense" (115), "Rewriting Indian history" (177) and "Restricting the Internet" (186). The majority of terms in these queries have very high number of postings, which suggests that they are either topic-neutral words (e.g., restrict, rewrite, transition), or they represent ideas or entities that were popular in newswire and governmental publications at the time (e.g., Internet, Indian). Moreover, these queries do not represent fixed phrases, i.e., that co-occur frequently in English language. Compare queries in this group to the query "Mad cow disease" (65), which performed very well. Although, the number of postings of individual terms is very high, the query represents a fixed expression, which occurs as a phrase in 213 documents.

Another reason of failure, which applies to both groups above, is over-stemming. We used Porter's stemmer with the strong stemming function in our searches. This function reduces various derivatives of the same lexeme to a common stem. For example, topic "Product customization" failed, because stems 'product' and 'custom' matched such words as 'production', 'productivity', 'customer', 'customs'. Strong stemming is

seen as a recall-enhancing technique. Weak stemming is likely to be more appropriate to the HARD task, as we are more interested in achieving high precision, rather than recall. Weak stemming keeps suffixes, and removes only endings, such as plural forms of nouns and past tense forms of verbs.

Another common reason for failure is that, some topic titles simply have insufficient information, for example in topic 186 ("Restricting the Internet"), the Description and Narrative sections narrow down the relevance criteria to the documents related to governmental restrictions of the Internet use in China.

4 Use of Statistical Association Measures for Noun Phrase Selection

In this and the following section we describe post-TREC experiments that we have conducted with the goal of better understanding the effect of noun phrases on retrieval performance.

In the phrase selection method, described in section 2, noun phrases, output by the noun phrase chunker, were ranked using the average idf of their constituent terms. This method is suitable for determining the informativeness of the individual words in the phrase, however it does not tell us whether the n-gram is a fixed multiword unit or a chance co-occurrence of words. In our HARD track experiments we noted that some of the phrases output by the chunker were, in fact, unsatisfactory chance word combinations.

We decided to explore the question whether the use of multiword units selected by Pointwise Mutual Information (Church et al. 1991) rather than any n-grams extracted by the phrase chunker in selecting terms for query expansion would result in better retrieval performance. We have conducted several experimental runs to address the above question, which are outlined below:

Run 1: All n-grams (n >= 2), output by the noun phrase chunker, were ranked by the average *idf* of their constituent terms. Single terms from the top *m* phrases were added to the original query terms from the topic title and searched using BM25 function.

Run 2: From each n-gram (n >= 2), output by the noun phrase chunker, we extracted all bigrams of adjacent words. For each bigram, *Pointwise Mutual Information* (*PMI*) was calculated as follows:

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} = \frac{f(x,y)N}{f(x)f(y)} \qquad (5)$$

Where:
f(x,y) - the number of documents in the corpus containing words *x* and *y* adjacent to each other and in the same order of occurrence;
f(x) and *f(y)* - the numbers of documents that contain words *x* and *y* respectively;
N – the number of documents in the corpus.

The reasons for using numbers of documents instead of word frequencies are pragmatic: numbers of documents are easily obtainable from the IR system Okapi, whereas calculating actual bigram frequencies is computationally expensive. We recognise, however, that the *PMI* score is more accurate when word frequencies are used.

N-grams were ranked by the highest *PMI* of their constituent bigrams. This is a rather crude selection method, but given the fact that we apply it to syntactically selected noun phrases, it is likely to produce satisfactory results.

Single terms from the top *m* n-grams were used in retrieval in the same manner as in run 1.

Run 3: PMI has limitations as a tool for selecting strongly associated bigrams, one of which is that it is biased towards low frequency words. Following Manning and Schuetze (1999), we used *I(x,y)*f(x,y)* for ranking bigrams in this run. The other parameters in this run are the same as in run 2.

Run 4: The same as run 1; all n-grams with n >= 1 are used.

Run 5: The same as run 4; n-grams containing no bigrams with PMI > 0 are removed.

In all runs the number of query expansion terms/phrases (*m*) was set to 30. The results are presented in table 2.

Run	Precision @ 10	Average Precision
Run 1	0.5220	0.2837
Run 2	0.5180	0.2730
Run 3	0.5220	0.2782
Run 4	0.4980	0.2805
Run 5	0.5200	0.2776

Table 2: Evaluation results

Top phrases ranked by average idf of their constituent terms	Top phrases ranked by the highest I(x,y)*f(x,y) of their constituent bigrams
VietCombank Ho Chi Minh City	Japanese yen
emotive topic	U.S dollar
15-nation bloc	VietCombank Ho Chi Minh City
VND-euro exchange rate	central bank
rollercoaster day	monetary policy
unified currency	exchange rate risks
HKFE's other Rolling Forex futures contracts	dollar euro exchange rate
Euro Transaction	exchange rate stability
Shorten Euro Transition Period Brussels	VND-euro exchange rate
third pillar	percentage point
BSS	15-nation bloc
Rolling Forex Euro Futures Contract	euro trades
tighter controls	oil pricing
euro zone	neutral stance
Own Single Currency	euro zone nations
euro bonds	euro zone
Monday's local newspaper De Morgen	currency traders
euro's launch	Monday's local newspaper De Morgen

Table 3: Top-ranked phrases (phrases in the shaded cells are selected by both methods).

The results do not provide any evidence that *PMI* is more useful than *idf* in selecting phrases for automatic query expansion. We have not tried other association measures, which may produce different results than *PMI*. Nevertheless, *PMI* or *I(x,y)*f(x,y)* can still be useful in selecting candidate query expansion phrases to be shown to the user in interactive query expansion. Table 3 shows top phrases ranked by *idf* , and *I(x,y)*f(x,y)* for the topic (180) "Euro Introduced".

5 Use of Phrases vs. Single Words in Search

Intuitively, the use of phrases, such as compound terms and proper names in search is expected to result in higher precision than the use of their constituent words separately.

We hypothesise that adjacent pairs of words, which have strong degree of association, will result in higher search precision when used in search as a phrase, as opposed to when used as single words.

To test this hypothesis we conducted an experiment, comparing two experimental retrieval runs against the baseline. The runs are described in more detail below.

Baseline run: All terms from TREC titles were used in search as single terms. BM25 search function was used to perform the search. This was exactly the same way we searched in the baseline run of HARD track.

Experimental run 1: All bigrams of adjacent words in TREC titles were extracted. For each bigram, Pointwise Mutual Information was calculated.

All bigrams with PMI > 0 are used in search as a phrase, i.e. using *Adjacency*[1] operator. Bigrams with PMI < 0 are split into individual words.

For example, in the topic title "Amusement Park Safety"

I (amusement, park) = 1.66
I (park, safety) = -7.6

The logical representation of the final query will be:

(amusement ***Adjacency*** park) ***BM25*** safety

Experimental run 2: The same as the experimental run 1 above, but all terms in the title are also added to the query as single terms, for example:

(amusement **Adjacency** park) **BM25** safety **BM25** amusement **BM25** park

The rationale behind including all terms into the query as single terms, is to relax the search criteria: if the phrase is rare, and retrieves only few documents, use of single terms will ensure that other documents which contain parts of the phrase will also be retrieved. Typically, phrases have quite high idf, therefore top retrieved documents are very likely to contain the phrases, used in the query.

Only 16 topic titles out of 50 had any bigrams with positive PMI. Both experimental runs had worse overall average precision than the baseline (see table 4). Only 2 topics in the experimental run 1 had better AveP than the baseline, whereas 7 topics in the experimental run 2 had better AveP than the baseline (see figure 2).

Run	Precision @ 10	Average precision
Single terms (baseline)	0.5240	0.3087
Bigrams + remaining single terms (experimental run 1)	0.5000	0.2819
Bigrams + all single terms (experimental run 2)	0.5180	0.3065

Table 4: Phrase search evaluation results

One of the reasons for this counter-intuitive result could be the fact that the bigrams may contain terms that are themselves in the query; either as single terms or as part of other bigrams. Robertson and his colleagues suggest that search term weighting should take into account the case of bigrams that have as their constituent terms single query terms, and propose a weighting scheme that adjusts their weights (Robertson et al. 2004). However, their method does not take into account the case of two or more bigrams that share a common term. We need further research to understand and deal with the complex case of bigrams containing other query terms, either, those which are part of other bigrams or exist as single terms in the query.

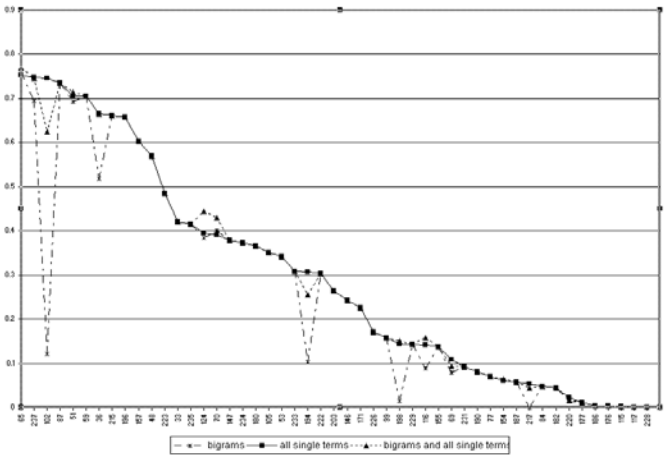


Figure 2: Results in average precision by topic of the two experimental runs and the baseline run.

6 Conclusions and Future Work

In this paper we presented a user-assisted search refinement technique, which consisted in showing to the user a list of noun phrases, extracted from the initial document set, and then expanding the query with the terms from the user-selected phrases.

[1] *Adjacency* is a pseudo-Boolean operator, which retrieves an unranked set of all documents, which contain the specified terms in adjacent positions in the same order as they were entered in the search statement.

The focus of our experiments in the HARD track of TREC-12 was on developing effective methods of gathering and utilising the user's relevance feedback. The evaluation results suggest that the expansion method overall is promising, demonstrating statistically significant performance improvement over the baseline run. More analysis needs to be done to determine the key factors influencing the performance of individual topics.

Post-TREC experiments conducted suggest that the use of PMI as a means of selecting n-grams for the purpose of term selection for query expansion is not promising. However, we should note that there was a number of simplifying assumptions in the use of PMI for the above purpose, which might have had a negative impact on its usefulness. It seems, however, possible that the use of PMI multiplied by the joint term frequency in selecting candidate query expansion phrases would result in a better selection of phrases to be shown to the user in interactive query expansion. We intend to experiment with alternative association measures to draw more strong conclusions about the usefulness of multiword units in IR.

After the official TREC results, we also conducted experiments to explore the use of phrases in searching. The results were not positive and confirmed the conclusions of the previous experiments reported in IR literature. We noted the difficulty in adjusting the weights of bigrams in searching when their constituent terms include other search terms.

Acknowledgements

This material is based on work supported in part by Natural Sciences and Engineering Research Council of Canada.

References

Allan, J. (2004). HARD Track Overview. Proceedings of the Twelfth Text Retrieval Conference, November 18-21, 2003, Gaithersburg, MD.

Beaulieu, M. (1997). Experiments with interfaces to support Query Expansion. Journal of Documentation, 53(1), pp. 8-19

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. Computational Linguistics, 21(4), pp. 543-565.

Church, K., Gale, W., Hanks, P. and Hindle D. (1991). Using statistics in lexical analysis. *In Lexical Acquisition: Using On-line Resources to Build a Lexicon*, ed. U. Zernik, Englewood Cliffs, NJ: Lawrence Elbraum Associates, pp. 115-164.

Koenemann, J. and Belkin, N. J. (1996). A case for interaction: a study of interactive information retrieval behavior and effectiveness. Proceedings of the Human Factors in Computing Systems Conference, Zurich, pp. 205-215.

Kwok, L. et al. (2004). TREC2003 Robust, HARD and QA track experiments using PIRCS. Proceedings of the Twelfth Text Retrieval Conference, November 18-21, 2003, Gaithersburg, MD.

Manning, C.D. and Schuetze, H. (1999). Foundations of Statistical Natural Language Processing, The MIT Press, Cambridge, Massachusetts, 1999.

Ramshaw, L. and Marcus, M. (1995). Text Chunking Using Transformation-Based Learning. Proceedings of the Third ACL Workshop on Very Large Corpora, MIT.

Robertson, S.E., Zaragoza, H. and Taylor, M. (2004). Microsoft Cambridge at TREC–12: HARD track. Proceedings of the Twelfth Text Retrieval Conference, November 18-21, 2003, Gaithersburg, MD.

Robertson, S.E., Walker, S. and Beaulieu, M. (2000). Experimentation as a way of life: Okapi at TREC. Information Processing and Management, 36, pp. 95-108.

Ruthven, I. (2003). Re-examining the potential effectiveness of interactive query expansion. Proceedings of the 26th ACM-SIGIR conference, Toronto, Canada, pp. 213-220.

Sparck Jones, K., Walker, S. and Robertson, S.E. (2000). A probabilistic model of information retrieval: development and comparative experiments. Information Processing and Management, 36(6), pp. 779-808 (Part 1); pp. 809-840 (Part 2).

Tombros, A. and Sanderson, M. (1998). Advantages of Query Biased Summaries in Information Retrieval. Proceedings of the 21st ACM SIGIR conference, Melbourne, Australia, pp. 2-10.

Vechtomova, O., Karamuftuoglu, M. and Lam, E. (2004). Interactive Search Refinement Techniques for HARD Tasks. Proceedings of the Twelfth Text Retrieval Conference, November 18-21, 2003, Gaithersburg, MD.

Comparative Evaluation of C-value in the Treatment of Nested Terms

Špela Vintar
Department of Translation
University of Ljubljana
Aškerčeva 2, SI – 1000 Ljubljana
spela.vintar@guest.arnes.si

Abstract

In statistical term extraction systems the identification and selection of nested term candidates often presents a challenge. The paper presents an implementation and evaluation of C-value, a heuristic that ranks and/or discards nested terms according to their stability in the corpus. The method was tested for English and Slovene, for both the overall performance of the term extractor improved using the proposed treatment of nested terms.

1. Introduction

For term extraction purposes, terms are essentially multi-word units with the additional property of "termhood". While nesting, i.e. the occurrence of smaller units within a larger lexical unit, is usually not the primary concern in collocation extraction for general language applications, it presents a core problem in term extraction systems, particularly in those based on statistical methods. A statistical term extractor should be able to, firstly, identify an instance of nesting and, secondly, decide which of the nested candidates are to be extracted.

In robust applications, as well as in several known commercial term extraction systems that involve little or no linguistic pre-processing, adequate treatment of nesting may be essential to the overall performance of the term extractor. If, for example, the trigram *axial compressive force* is extracted from an untagged corpus, the system will most probably detect also the statistical relevance of the bigrams *axial compressive* and *compressive force*. While the latter may indeed be terminologically relevant, the former is clearly noise and does not correspond to any of the typical terminological patterns in English.

Even in systems which use morphosyntactic analysis and extract, for example, noun phrases, the ranking of nested terms may not be an easy task. Consider for example the collocation *reactor coolant system replacement* and the nested bi- and trigrams it contains. Should *reactor coolant* and *coolant system* both be retained, while *system replacement* rejected? Is the extracted fourgram at all terminologically relevant? Clearly such decisions depend on the purpose of the term extraction task and on the requirements of the target users.

We report on an implementation of C-value for the treatment of nested terms in a bilingual term extraction system. The system was designed for English and Slovene in two parallel versions: the statistical version which works with raw texts and no morphosyntactic tagging to extract term candidates monolingually and then identify translation equivalents, while the hybrid version uses tagged corpora and syntactic term patterns.

The focus of the experiments reported here is on statistical collocation extraction and the evaluation of the term ranking achieved through the treatment of nesting. First we briefly describe the term extraction method used in both versions of the system, then we present the treatment of nesting and finally compare results obtained for English and Slovene. The last section gives a discussion of the approach together with some ideas for improvement.

2. Related Work

One method for ranking nested collocations is C-value (Frantzi & Ananiadou, 1996). The method is a purely statistical measure of the stability of a collocation in a given corpus. If a candidate string is not found as nested, the C-value is calculated from its total frequency and length. If it is found as nested, the C-value is calculated from its total frequency, length, frequency as a nested string, and the number of longer candidate terms it appears in. C-value was later supplemented by the authors and used in combination with morphologically processed data (Frantzi&Ananiadou, 1999). We use a slightly modified version of C-value as described by Nakagawa&Mori (1998).

That NLP applications for Slavic languages heavily depend on morphological processing is a long known fact. Nenadić et al. (2003) for example explore morpho-syntactic clues for the terminological processing of Serbian. It is however our belief that there is – and will be for some time – still need for robust, statistical applications able to handle a wide variety of languages. Term extractors are for example incorporated into Translation Memory systems, where term candidates are identified largely on the basis of frequency and existing term banks, independently of the language pair.

An alternative method for the resolution of nesting, or rather collocation extraction with an integrated ranking of nestings, was proposed by Silva et al. (1999). It uses

LocalMax, an algorithm that measures the "glue" between words and proposes possible term boundaries.

It is, however, our observation that many term extraction methods involve no explicit strategy to tackle this problem, which probably means that in cases of nesting no additional ranking is performed and all variants are extracted if they fulfil other criteria of *termhood*.

3. Term Extraction Methods

In the statistical version of the term extraction system, the measure used to extract collocations is the Log-Likelihood Ratio (LL), which had been identified as a reliable indicator of terminologically relevant collocations by various authors (Dunning, 1993). LL applies well to special purpose corpora because it will extract rare collocations and is more appropriate for sparse data. The system is currently tuned to extract contiguous collocations of length 2-6 units.

To determine the *termhood* of a candidate collocation, we use the Term Frequency - Inverse Document Frequency (*tf.idf*) of single words. Although this measure of term weighting is traditionally used in Information Retrieval, is works well for term extraction.

$$tf.idf(i,j) = \begin{cases} (1+\log(tf_{i,j}))\log\dfrac{N}{df_i}, & tf_{i,j} \geq 1 \\ 0, & tf = 0 \end{cases}$$

where N is the number of all documents in a collection, $tf_{i,j}$ is the frequency of the term w_i in the document collection d_j and the document frequency df_i is the number of documents where the term occurs at least once. The *tf.idf* method is a useful measure of *termhood* only in cases where the corpus for term extraction, or the document collection, is heterogeneous. If the method were applied to a special domain corpus all general terms of the domain would have been missed.

The hybrid version of the term extractor was designed to extract predefined tag patterns regardless of their frequency in the corpus. The patterns were determined in co-operation with domain experts on the basis of observed sequences of part-of-speech tags in the corpus. For Slovene, instead of just part-of-speech it is useful to include other relevant grammatical categories which help identify term phrases. For example, case information is crucial in detecting noun phrase boundaries.

Because nesting is primarily the problem of statistical term extractors, we will limit the results presented to those obtained without morphological information or syntax patterns.

Some results presented by the statistical term extraction system before the treatment of nesting are presented in Table 1.

Candidate term	**C-value**
Economic Zone	79.92
Economic operators	29.92
Election Day	219.92
Environmental Fund	29.92
Environmental Impact	49.92
Environmental Protection	309.92
Environmental Protection Program	219.85
European Commission	109.92
European Communities	339.92
European Union	79.93
European Union hereinafter	59.86
European standards	29.93

Table 1: Examples of English extracted term candidates

4. Implementation of C-value

C-value is a well-known, but not as widely used method of resolving nested collocations (Frantzi/Ananiadou 1996, Nakagawa/Mori 1998), which ranks terms according to their stability in the corpus. C-value is defined as:

$$C-value(a) = (length(a)-1)(freq(a) - \frac{t(a)}{c(a)})$$

where "a" is a collocation, t(a) is the frequency of "a" in longer candidates of collocations and c(a) is the number of longer candidates of collocations including "a". If frequencies are low, the value may be negative, so that for more transparent results it might be advisable to multiply the frequencies accordingly.

Looking at the examples in Table 1 we see that C-value for longer, but less stable or less relevant collocations, like *European Union hereinafter*, is indeed smaller. The highest C-score in our corpus was assigned to *Bank of Slovenia*, a good and terminologically relevant collocation which might compete with *the Bank of Slovenia, Bank of Slovenia shall* or *the Bank of*. At this point we must give a short explanation of the way C-value weights collocations. If the longer phrase is equally frequent and the shorter phrase occurs with no other modifiers, as is the case with *the Bank of Slovenia*, the longer collocation will be assigned a higher score, as can be seen from the formula. For this reason, constant modifiers such as articles or fixed pronouns must be dealt with in some other way, e.g. by employing a stopword filter over the candidate list.

C-value helps select the most relevant candidate from a list of related collocations. Because the score relies heavily on the frequency of the phrase, we cannot define a general threshold to filter out nested phrases. It should also be noted that although the original implementation of C-value suggests its efficiency also as a term weighting method, *termhood* cannot really be determined solely on the basis of length and frequency.

When dealing with Slovene, absence of morphological processing will invariably have a negative effect on term extraction in general, and especially on the calculation of C-value. Consider the following example:

Candidate term	C-value
Ekonomska cona	61.87
Ekonomske cone	71.87
Ekonomske cone Koper	63.75
Ekonomske cone Maribor	63.75

Table 2: Example from Slovene

Because of case inflections, the base form *Ekonomska cona* receives a lower score than its two obviously inferior variations, *Ekonomske cone Koper* and *Ekonomske cone Maribor*. Also, the score of the Slovene term is lower than the non-inflected English equivalent Economic Zone (see Table 1).

Nevertheless, the method proved extremely useful in discarding syntactically incomplete collocations, for example Adjective + Adjective or Noun + Adjective, where the final noun was missing.

Candidate term	C-value
Okrajna volilna	61.89
Okrajna volilna komisija	**123.79**
Opravljanje nadzora	31.89
Opravljanje zavarovalnih	71.89
Opravljanje zavarovalnih poslov	**143.80**

Table 3: Example with resolved nesting

5. Results

Candidate lists were produced with both versions of the term extractor, statistical and hybrid, and evaluated by a professional terminologist. Although the system extracted bilingual term pairs from a parallel corpus, evaluation was performed only for the Slovene part, because Slovene was the mother tongue of the evaluator. The statistical system achieved 0.49 precision at 0.48 recall after the treatment of nesting, and the linguistic version 0.51 precision at 0.65 recall. To evaluate the impact of the treatment of nesting on the system, we compared the candidate lists before and after the treatment of nesting. The former would have achieved only 0.31 precision with an insignificantly higher recall of 0.52.

	Stat.-before	Stat.-after	Ling.
Precision	0.31	0.49	0.51
Recall	0.52	0.48	0.65

Table 4: System evaluation

For English, the evaluation of the treatment of nesting showed that the English candidate list of 2706 items contained 487 or 18% nestings, of which 81% were correctly identified and removed.

It should perhaps be noted that the external evaluator sometimes selected both term variants, the nested shorter collocation and the longer collocation. This is perfectly logical considering the nature of terminological work and the multifarious purposes of terminological resources. Therefore, for some applications nested terms represent no noise and thus call for no special ranking.

6. Discussion

Overall results of the system may seem rather poor, however the evaluation methodology was extremely strict in the sense that it allowed only candidates of a very high quality and terminological significance.

An important disadvantage of this method is its incompleteness, i.e. the limitation of collocation length. In itself there is no limit, however the formula contains the factor defined as "number of larger units containing *a*". Clearly, when we extract collocations we impose some sort of length limit, meaning that for the longest units C-value will be 0 because no larger units will contain them.

As mentioned above, the efficiency of the system depends highly on some prerequisites, for example that collocations containing articles, pronouns, numerals or non-lexical verbs at the beginning or end must be processed beforehand using either a stopword filter or a threshold of term relevance, such as *tf.idf*.

Furthermore, some issues concerning term nesting remain subject to discussion as to whether or not they depend on the domain and the target user (see also Estopa 1999). Considering examples such as *steam generator replacement project* we see that the longer collocation is highly significant in a given context, but may be irrelevant for the domain of nuclear engineering in general. The latter, especially for the purposes of traditional terminography, would rather include terms like *steam generator*. We argue that C-value can be used as a valuable tuning parameter in adjusting a term extractor to the user's requirements.

7. Conclusion

The paper presented an implementation of C-value for selecting and discarding nested term variants from term candidate lists produced with a statistical term extraction system. Results show that the method improves the performance of the system both for English and Slovene. Since the method is useful primarily in statistical term extraction, it would have been beside the point to propose moprhosyntactic processing as a path for future improvements. Rather, for inflected languages it might be useful to include string-based merging of word forms, which would enable a more accurate calculation of the C-value.

References

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational linguistics,* 19:61-74.

Estopa, R. (1999). Extracció de terminologia: elements per a la construcció d' un SEACUSE (Sistema d' Extracció Automatica de Candidats a Unitats de Significació Especializada). PhD Thesis, Universidad Pompeu Fabra, Barcelona.

Frantzi, K.T. and Ananiadou, S. (1996). Extracting nested collocations. In: 16thConference on Computational Linguistics, COLING, p. 41-46.

Frantzi, K.T. and Ananiadou, S. (1999). The C-Value/NC-Value domain independent method for multi-word term extraction. *Journal of Natural Language Processing,* 6(3): 145-179.

Maynard, D. and Ananiadou, S. (2000) Terminological Acquaintance: the importance of contextual information in terminology. Proc. of NLP2000 Workshop on Computational Terminology for Medical and Biological Applications", Patras, Greece, 2000, pp. 19-28.

Nakagawa, H. and Mori, T. (1998). Nested Collocation and Compound Noun for Term Extraction. In: Computerm '98, First Workshop on Computational Terminology, p. 64-71.

Nenadić, G., Spasić, I. and Ananiadou, S. (2003). Morpho-Syntactic Clues for Terminological Processing in Serbian, *Proceedings of EACL Workshop on Morphological Processing of Slavic Languages*, Budapest, Hungary.

Silva, J. F.; Dias, G.; Guilloré, S. ; Lopes, J.G.P (1999): Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units. Actes 9th Portuguese Conference in Artificial Intelligence, Springer-Verlag.

Thank You!

Workshop Organisers

Gaël Harry Dias (Beira Interior University, Portugal)

José Gabriel Pereira Lopes (New University of Lisbon, Portugal)

Špela Vintar (University of Ljubljana, Slovenia)

Workshop Local Organizing Committee

João Paulo Cordeiro (Beira Interior University, Portugal)

Sara C. Madeira (Beira Interior University, Portugal)

Sérgio Nunes (Beira Interior University, Portugal)