Integer Linear Programming for Dutch Sentence Compression

Jan De Belder and Marie-Francine Moens

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
jan.debelder@cs.kuleuven.be, sien.moens@cs.kuleuven.be

Abstract. Sentence compression is a valuable task in the framework of text summarization. In this paper we compress sentences from news articles from Dutch and Flemish newspapers written in Dutch using an integer linear programming approach. We rely on the Alpino parser available for Dutch and on the Latent Words Language Model. We demonstrate that the integer linear programming approach yields good results for compressing Dutch sentences, despite the large freedom in word order.

1 Introduction

Since the end of the 20th century, the compression of texts has been an active research topic in natural language processing (see for example the Document Understanding Conferences [1], and the more recent Text Analysis Conferences [2]). As this is a very difficult problem, it has often been reduced to the summarization of individual sentences, commonly referred to as sentence reduction [3] or sentence compression. This summarization task is the easiest in a word deletion setting, where we remove words from the original sentence, while maintaining a grammatical and coherent sentence that conveys the most important information [4]. For the Dutch language, the research in this area is limited. There has been some work on the summarization of documents in [5]. The compression of individual sentences has only been approached from a subtitle generation viewpoint [6] [7] [8] and a headline generation viewpoint [9]. In this paper, we investigate a generic method for sentence reduction, based on integer linear programming [10]. Required for this method are a language model, a parser, and a integer linear programming (ILP) solver.

The ILP approach operates by viewing sentence compression explicitly as an optimization problem. With a binary decision variable for each word in the original sentence, indicating whether or not it should be in the compressed sentence, the ILP solver finds an assignment for these variables that maximizes the probability of the sentence in the language model. In order to create well-formed summary sentences, the compression model might incorporate additional constraints that use grammatical rules of the language. As the most interesting information is most likely not very prominent in the language model, there is also need for a way of incorporating this information in the compressions. This is the function of the significance model.

In the next section we give an overview of relevant background work. Section 3 shortly introduces the tools we used for Dutch. Section 4 describes the main ideas of the integer linear programming approach. Our experimental setup can be found in section 5, and section 6 reports on the results. Finally, we give our conclusions and indications for future work in section 7.

2 Background

Summarization or compression of text is a useful, but non-trivial application of natural language processing. Currently, there are several settings being researched that include the summarization of single documents [11], the summarization of multiple documents [12], and the summarization of single sentences. In this paper, we address the last setting.

Nearly all approaches of sentence compression rely on word deletion in such a way that the result is still a grammatical sentence, and conveys the most important information of the original sentence. A common application is headline generation based on the content of a larger text. By looking for headlines that are a subsequence of words in the first sentence of a news article, a sentence compression corpus can automatically be constructed. The offset for this approach was given in [4]. These authors used a parallel corpus of compressed and original sentences based on the Ziff-Davis corpus of news articles in the computer technology domain. The authors evaluated two compression methods. A noisy channel model considers an original sentence as the compressed sentence to which noise has been added. It assigns the most likely compression to the full sentence using Bayes rule, where the probability of a noisy component given a summary sentence is learned from the training data. The decision based model learns the discriminative reductions of the parse tree with a decision-tree learner based on the training data. The noisy-channel model is, however, not directly applicable for Dutch, due to lack of a Probabilistic Context Free Grammar. The decision based model has the disadvantage that the desired amount of compression cannot be given as a parameter.

In [9], headline generation was studied for the Dutch language. The method takes inspiration from the linguistically motivated Hegde trimmer algorithm [13], which employs rules to reduce the parse tree of a sentence, but learns the rules automatically using Transformation Based Learning, an error-driven approach for learning an ordered set of rules. The corpus that was used originates from Dutch news articles with matched headlines, taken from the Twente News Corpus.

Another setting in the compression of single sentences is the generation of subtitles for broadcasts. This is the case that has been mostly studied for Dutch [6] [7] [8]. These methods are based on shallow parsing and most of them require a parallel corpus for training. However, recent work [14] has shown that a word deletion approach is not very suited for subtitle generation.

There are also a few unsupervised approaches for sentence compression. [15] summarize the transcription of a spoken sentence, given a fixed compression rate. They use dynamic programming to find an optimal scoring solution, that takes a language model and the confidence of the speech recognizer into account. [16] define a semi-supervised and unsupervised version of the noisy channel model of [4]. [10] use an integer linear programming approach, which is applicable for any language, given the availability of a parser. This is the method that we will discuss, use, and modify in the remainder of this paper.

3 Language Tools

In this section we describe the tools we used for constructing our Dutch sentence compression system.

3.1 Parsing

For parsing the Dutch sentences, we use the Alpino parser [17]. The Alpino system is a linguistically motivated, wide-coverage grammar and parser for Dutch in the tradition of HPSG. It consists of about 800 grammar rules and a large lexicon of over 300,000 lexemes and various rules to recognize special constructs such as named entities, temporal expressions, etc. The aim of Alpino is to provide computational analysis of Dutch with coverage and accuracy comparable to state-of-the-art parsers for English. It is freely available for download.[1]

3.2 Latent Words Language Model

The Latent Words Language Model (LWLM) models the contextual meaning of words in natural language as latent variables in a Bayesian network [18]. In a training phase the model learns for every word a probabilistic set of synonyms and related words (i.e. the latent words) from a large, unlabeled training corpus. During the inference phase the model is applied to a previously unseen text and estimates for every word the synonyms for this word that are relevant in this particular context. The latent words help to solve the sparsity problem encountered with traditional n-gram models, leading to a higher quality language model, in terms of perplexity reduction on previously unseen texts [19]. In this article the model is trained on a 25m token corpus, consisting of Dutch newspaper articles.

4 Integer Linear Programming Approach to Sentence Compression

In this section we will lay out the sentence compression method based on integer linear programming, following the line of work in [10]. We will start by

[1] http://www.let.rug.nl/vannoord/alp/Alpino/

shortly explaining what integer programming is, and how the basic method works by maximizing a language model probability. There are also extra constraints needed to make sure that a meaningful and grammatical sentence is obtained. In section 4.4 we discuss the significance model, that ensures that the generated compressions also contain topics of interest.

4.1 Integer Linear Programming

Integer linear programming is a restricted case of linear programming, where the values of the variables are limited to be only integers, instead of any real number. Linear programming tries to maximize (or minimize) an objective function, by searching for optimal values for the variables that constitute the objective function. This objective function is a linear combination of these variables, hence the name. The finding of an optimal combination of values is usually constrained. These constraints ensure that the variables cannot be infinitely large, and that the value of one variable can influence the other variables.

Integer programming has been used often in Natural Language Processing, for many different tasks. In many situations, NLP constitutes searching in very large hypothesis spaces, like packed forests of parse trees [20]. Other applications include a.o. coreference resolution [21] and semantic role labeling [22]. Integer linear programming, a technique that has often been used in optimalisation theory for many decades, is very well suited for these kind of problems, as it enables us to efficiently search for the optimal solution, and at the same time incorporate constraints on a global scale.

4.2 Integer Programming for Sentence Compression

Given a sentence $W = w_1 \dots w_n$, our goal is to obtain a sentence W^*, with a reduced number of words. For a sentence $W = w_1 \dots w_n$, we first need decision variables to indicate whether or not w_i should be in the compressed sentence. We notate these variables with y_i, with a value of 1 if word w_i is in the compressed sentence, and 0 if it is not. For clarity, suppose we want the ILP solver to find a sentence that maximizes a unigram model, then the objective function would look like this:

$$max\ z = \sum_{i=1}^{n} y_i P(w_i),$$

with $P(w_i)$ being the unigram probabilities. This overly simple model is not adequate; a trigram model would have much better performance. This comes down to adding three additional types of variables. In short, we need n extra variables to indicate whether or not a word starts the sentence (p_i), and $\frac{n \cdot (n-1)}{2}$ decision variables that indicate whether two words end the sentence (q_{ij}). Finally, there are $\frac{n \cdot (n-1) \cdot (n-2)}{6}$ variables needed to indicate whether a specific trigram $w_i w_j w_k$ is in the sentence (x_{ijk}). These three types of variables are needed for constraints on the language model. For example, only one word can start the sentence, which translates to a constraint in the ILP model. Without these constraints, the ILP would set all variables to 1, and say that all words start the sentence. The complete list of constraints can be found in [10], but will not be repeated due to spatial constraints, and the fact that they are not required to understand to operations behind the method. The objective function of the integer linear programming problem is given in the following equation[2]:

$$\begin{aligned} max\ z = & \sum_{i=1}^{n} p_i P(w_i|\text{start}) \\ & + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} x_{ijk} P(w_k|w_i w_j) \\ & + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} P(\text{end}|w_i w_j) \end{aligned} \tag{1}$$

4.3 Linguistic Constraints

The ILP model given above is language independent. However, the fact that a sentence has a high probability in a language model, does not make it a grammatical and fluent sentence. That is why there is the need to incorporate language specific grammatical information in the method.

The constraints described below are motivated from a linguistic and intuitive point of view and are often dependent on the language used. These constraints are based on a parse tree and the grammatical relations of a sentence, and can be used in combination with any parser. In [10], the Robust Accurate Statistical Parsing toolkit was used [23]. As described in section 3.1, for Dutch we are limited to the use of Alpino.

Modifier Constraints It is often the case that determiners can be left out of the compression (especially in the case of headline generation). This still yields a grammatical sentence. The other way around, i.e. keeping the determiner but removing its head word, is not acceptable. This leads to the following constraint:

$$\begin{gathered} y_i - y_j \geq 0 \\ \forall i, j : w_j \in w_i\text{'s determiners} \end{gathered} \tag{2}$$

If a determiner w_j is in the compression, which corresponds to y_j having the value 1, the constraints force y_i to take the value 1 as well, causing the head word w_i to be in the compression.

Some determiners cannot be left out, especially when they change the meaning of their head word, and thus probably the meaning of the entire sentence. The most trivial one is the word ‘not’. We also included the word ‘none’. An important modifier for Dutch is the word *er*, which translates roughly as ‘there’[3].

[2] From here on we assume all probabilities are log-transformed

[3] For example in the sentence ‘*Er is melk in de koelkast*’, which translates to ‘There is milk in the fridge’. Sometimes this is not as clear. The sentence ‘Something has

This constraint can be removed, but it generates more fluent sentences, rather than headline-style sentences. Possessive modifiers are also added to the list.

$$\begin{aligned} & y_i - y_j = 0 \qquad (3) \\ & \forall i, j : w_j \in w_i\text{'s determiners} \wedge \\ & \qquad w_j \in (\text{not, none, possessives, '}\textit{er}\text{'}) \end{aligned}$$

Note that the difference between constraints 2 and 3 is in the sign of the equation: constraint 2 uses a $\geq$ sign to indicate that w_i can be in the compression by itself, but w_j can not. Constraint 3 uses an $=$ sign, which mean that either both w_i and w_j have to be in the compression or either none of them can be in the compression.

Argument Structure Constraints The next constraints are needed for the overall sentence structure. Constraint 4 makes sure that if there is a verb in the compressed sentence, then so must be its arguments. The reverse also has to be true: if there is a subject from the original sentence taken for the compressed sentence, so must be the corresponding verb.

$$\begin{aligned} & y_i - y_j = 0 \qquad (4) \\ & \forall i, j : w_j \in \text{subject/object of verb } w_i \\ & \sum_{i:w_i \in verbs} y_i \geq 1 \qquad (5) \end{aligned}$$

Constraint 5 requires that, if there is a verb in the original sentence, there should also be at least one in the compressed sentence.

One of the peculiarities of Dutch[4] are separable verbs that fall apart into their original parts, when you conjugate them. For example, *toepassen* (to apply), becomes in the first person singular *ik pas toe* (I apply). If a compressed sentence contains the stem of the separable verb, it should also include the separated part, and vice versa. The parser detects these separable verbs, so we can define the following constraint:

$$\begin{aligned} & y_i - y_j = 0 \qquad (6) \\ & \forall i, j : w_j = \text{separated part of separable verb } w_i \end{aligned}$$

Furthermore we also require the predicative adjectives to be included together with their head, and the same for reflexive objects such as 'themselves'.

There are two other constraints needed for prepositional phrases and subordinate clauses in order to ensure that the introducing term is included, if any word from the phrase or clause are included (defined in equation 7). Subordinate

to be done' translates to '*Er moet (has) iets (something) gedaan (done) worden (to be)*'.

[4] This is also common in German and Hungarian.

clauses are those that begin with a wh-word, or with subordinating conjunctions such as ‘after’ or ‘because’. The reverse should also hold (see equation 8).

$$
\begin{aligned}
y_i - y_j &\geq 0 && (7)\\
\forall i, j : w_j &\in \text{PP/SUB} \ \wedge \\
w_i &\quad \text{starts PP/SUB} \\
\sum_{i: w_i \in \text{PP/SUB}} y_i - y_j &\geq 0 && (8)\\
\forall j : w_j \text{ starts PP/SUB}
\end{aligned}
$$

General Constraints Alpino is able to detect multi word units (MWUs). These can be names of persons, such as *Minister Van Der Donck*, but also parts of expressions, such as *op wacht staan* (to stand guard). For simplicity we define a constraint that either all words of the MWU should be included, or none of them.

Related to the compression length, it is possible to define an upper and lower bound on the generated compression. Enforcing a length of at least l tokens is done with the following constraint:

$$\sum_{i=1}^{n} y_i \geq l \tag{9}$$

Defining an upper bound can easily be done by replacing the $\geq$ sign with $\leq$.

4.4 Significance Model

A probable side effect of relying on a language model to generate compressions, is that the model will prefer known words. This has as a consequence that the most important words in the sentence, for example names of persons, will not be likely to appear in the compression. The solution for this problem lies in a significance model. This model assigns a weight to every topic word in the sentence, with a topic word being a noun or a verb. The weights are based on several statistics, and calculated with the following equation:

$$I(w_i) = \frac{l}{N} f_i log \frac{F_a}{F_i} \tag{10}$$

where f_i and F_i are the frequencies of word w_i in the document and a large corpus respectively, F_a the sum of all topic words in the corpus. l is based on the level of embedding of w_i: it is the number of clause constituents above w_i, with N being the deepest level in the sentence . To incorporate these weights in the objective function given by equation 1, the sum of equation 10 over the topic words can be simply added, resulting in the following equation:

$$\begin{aligned} max\ z = {} & \lambda \sum_{i=1}^{n} y_i I(w_i) + \sum_{i=1}^{n} p_i P(w_i|\text{start}) \\ & + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} x_{ijk} P(w_k|w_i w_j) \\ & + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} P(\text{end}|w_i w_j) \end{aligned} \tag{11}$$

The parameter λ weighs the importance of the language model versus the significance model, and can be estimated on a small set of training data.

5 Evaluation

5.1 Data

The data consists of news articles written in Dutch, coming from major Belgian and Dutch newspapers and crawled from the Web pages of the news providers. We selected the websites and articles at random, to have a diverse set of texts. The articles date back to the beginning of 2008. We used a set of articles from 31/1/2008 and 1/2/2008 for development and training, and articles from 6/2/2008 and 8/2/2008 for the evaluation.[5] We manually segmented the articles into sentences, to ensure a clean dataset. The training and development data consisted of 40 articles, the evaluation data of 30.

Since the evaluation is done manually, as will be described in section 5.3, the amount of sentences that we can evaluate is limited. Here we took the first sentence of each article in the evaluation set, and limited these further to sentences that contain at least 15 tokens. This resulted in a set of 21 sentences, with an average length of 20.8 tokens, ranging over a diverse set of topics.

We used a different data set to train the Latent Words Language model. We took a 25 million token subset of the Twente News Corpus [24], from four different newspapers in the year 2005. The dictionary size was limited to 65.000 words. We also used this data to estimate the corpus frequency of the topic words, as described in equation 10. If a topic word was not present in the corpus, we estimated its weight as the average of the other topic words in the sentence.

5.2 Systems

For the evaluation we tested the system in four different settings, all based on the integer linear programming approach. The first system relies solely on the language model, and does not use any grammatical information. The second

[5] This difference in time was needed to ensure that no articles in the evaluation data overlapped with those in the development data.

system does use the grammatical constraints. The third and fourth system both add the significance model, but with different values for the parameter λ. As described in section 4.4, this parameter weighs the importance of the significance model against the language model. During initial testing it became clear that it is very difficult to estimate this parameter. Values that work for some sentences yield lesser results on other sentences. It also has a significant influence on the length of the compression, where higher values for λ tend to generate longer sentences. Higher values cause the system to only include the topic words, while still being limited by the constraints, which results in using all the topic words without everything that is dictated by the constraints. For these reasons, we did the evaluation with two different values for λ: 0.75 and 1.5, that both had good empirical results on the development data.

Finally, we constrained the systems to generate compressions of at least 40% of the original length, by using the constraint in equation 9.

5.3 Evaluation

As is good practice in the testing of summarization systems, we opted for manual evaluation. We did two different experiments. In the first experiment, we presented the participants with a list of generated compressions, each from a different original sentence. We asked the participants to give a score for the grammaticality of each sentence, on a five point scale. In the second experiment the participants were given the original sentences together with the corresponding compressions, and they were asked to rate the compressions based on the retention of the most important information, again on a five point scale. The sets of sentences were generated at random: each set contained compressions from the different systems. Together with the four systems defined above, we added a manually constructed set of compressions made by one of the authors. The participants were told that all the sentences were machine generated. This allows us to compare the machine generated compressions with one made by a human, and define an upper bound on the performance that is achievable in a word-deletion setting. In total we had 15 participants, each grading 21 sentences based on grammaticality, and another 21 sentences on content.

Using the manually constructed set of compressions, we also calculated the ROUGE scores [25], as often applied in the DUC competitions. We used the ROUGE-2, ROUGE-L, and ROUGE-SU4 metrics, that assign scores based on bigram co-occurrences, the longest common subsequence, and skip-bigrams in combination with unigrams respectively.

6 Results

6.1 Human Evaluation

Grammaticality The results on the manual evaluation can be found in table 1. From the column that reports on the grammaticality of compressions, it is clear that the grammatical constraints are necessary. The system that uses only the language model to generate compressions, did not came up with many meaningful sentences. This is very likely due to the limited size of the language model used. The systems that do use the grammatical constraints usually come up with a grammatical compression. The median of grammaticality scores is 4, for each of the three systems that used the grammatical constraints. Annotators often punished the compressions due to not incorporating the determiners, which generates more headline-like compressions. The leaving out of commas was also a cause for lower ratings. In one case none of the systems was able to include the main verb and subject, which did not happen when using a longer minimum compression length. The biggest problem is the needed inversion of a verb and a subject when a prepositional phrase is removed from the beginning of the sentence. Switching the verb and the subject in a sentence would require substantial modifications to the ILP method. The grammatical information from the parse tree would not just lead to the adding of more constraints, but to the addition of more decision variables and a modification of the objective function, which we leave for further research.

Significance Model Looking further we can see that the significance model has an impact on the information retention, although this is rather limited. Despite the fact that the last system ($\lambda = 1.5$) generates on average sentences that are 14% longer, this has little influence on the scores given by the participants of the experiment. The reason for this is that the most important information usually takes the role of subject or object, and is thus already required to be in the compression. The difference in score between the best system and the human made compressions is larger than for the grammaticality, but it should be noted that the human made compressions are on average almost 10% longer.

System	Avg. Comp. Rate[6]	Grammar	Information
Human	66.9%	4.71 ± 0.40	4.43 ± 0.53
LWLM	43.0%	1.29 ± 0.54	1.26 ± 0.30
LWLM+Gram	43.3%	3.45 ± 1.47	3.14 ± 1.31
LWLM+Gram+Sig ($\lambda = .75$)	49.0%	3.81 ± 1.38	3.19 ± 1.67
LWLM+Gram+Sig ($\lambda = 1.5$)	57.5%	3.98 ± 1.12	3.41 ± 1.19

Table 1. Manual evaluation results of the four systems and the handcrafted summaries, on grammaticality and information retention of the compressions.

6.2 Automatic Evaluation

From the results in table 2 we can conclude that the automatic evaluation measures all follow the human judgment. The version of the system with the signifi-

[6] We define the average compressed rate as the average percentage of words retained in the compression.

cance model ($\lambda = 1.5$) scores the best, which indicates that this model generates compressed sentences that are the closest to the handcrafted summaries.

6.3 Discussion

In general, the method performs rather well. When compared to the human made summaries, the best model only scores ± 1 point lower, both on grammaticality and content. We also tested whether the human made summaries were possible to create by the ILP method, using the grammatical constraints imposed. In 12 out of the 21 cases, this was not possible. Often the cause was a small error in the parsing process, especially in the case of PP-attachments.

Another related problem can be found in the compression of names. Often these are accompanied by a description of their function, for example 'The French president Sarkozy'. Without loss of information, this can easily be reduced to 'Sarkozy'. But when talking about the Serbian president Boris Tadić, the participants of the experiments preferred the descriptive compression 'the Serbian president' over the actual name 'Boris Tadić'. This problem is not only present in Dutch, but in summarization in general.

In these experiments we defined specific values for λ and a specific lower bound on the sentence length, in order to obtain just one compression from every system. However, the systems can easily generate an entire set of compressions by varying the parameters, more often than not generating better compressions than given here. As the solving of the ILP problem is several orders of magnitude faster than parsing the sentence with the Alpino parser, it is our opinion that the determination of the best compression, given a set of possible compressions, can better be handled in a later stage.

System	ROUGE-2	ROUGE-L	ROUGE-SU4
LWLM	0.240	0.569	0.341
LWLM+Gram	0.431	0.650	0.469
LWLM+Gram+Sig ($\lambda = .75$)	0.472	0.697	0.505
LWLM+Gram+Sig ($\lambda = 1.5$)	0.508	0.712	0.530

Table 2. Automatic evaluation results with the ROUGE toolkit, using the handcrafted summaries as a gold standard.

7 Conclusion

In this paper we have presented a sentence compression method for Dutch, a free word order language. We used an integer linear programming approach that finds a compression by maximizing the language model probability, while constrained to be grammatical. For this we used Alpino, a parser for Dutch, and the Latent Words Language Model. We needed extra language-specific constraints on the generated compressions to maintain the meaning, which we accomplished by using the output of the parser. We also identified some shortcomings, by checking whether the handcrafted compressions can be generated under the grammatical constraints, which was not always the case.

The next step is to extend the integer linear programming approach to allow for words to swap places, allowing the model to generate more grammatical compressions. We also believe that the meaningful compression of person names with their description could be learned from training data, in addition to this otherwise unsupervised method.

Acknowledgments

This research is funded by the Dutch-Flemish NTU/STEVIN project *DAISY*[7] (ST 07 015) and the EU project *PuppyIR*[8] (EU FP7 231507).

References

1. Nenkova, A.: Automatic text summarization of newswire: Lessons learned from the document understanding conference. In: Proceedings of the National Conference on Artificial Intelligence. Volume 20., MIT Press (2005) 1436
2. Text Analysis Conference (TAC). http://www.nist.gov/tac/
3. Jing, H.: Sentence reduction for automatic text summarization. In: Proceedings of the 6th Applied Natural Language Processing Conference. (2000) 310–315
4. Knight, K., Marcu, D.: Statistics-based summarization-step one: Sentence compression. In: Proceedings of the National Conference on Artificial Intelligence, MIT Press (2000) 703–710
5. Angheluta, R., De Busser, R., Moens, M.F.: The use of topic segmentation for automatic summarization. In: Proceedings of the ACL-2002 Workshop on Automatic Summarization, Citeseer (2002)
6. Vandeghinste, V., Pan, Y.: Sentence compression for automated subtitling: A hybrid approach. In: Proceedings of the ACL Workshop on Text Summarization. (2004) 89–95
7. Vandeghinste, V., Sang, E.: Using a parallel transcript/subtitle corpus for sentence compression. In: Proceedings of LREC 2004, Citeseer (2004)
8. Daelemans, W., Hothker, A., Sang, E.: Automatic sentence simplification for subtitling in Dutch and English. In: Proceedings of the 4th International Conference on Language Resources and Evaluation, Citeseer (2004) 1045–1048
9. de Kok, D.: Headline generation for Dutch newspaper articles through transformation-based learning. Master's thesis
10. Clarke, J., Lapata, M.: Global inference for sentence compression: An integer linear programming approach. Journal of Artificial Intelligence Research **31**(1) (2008) 399–429
11. Teng, Z., Liu, Y., Ren, F., Tsuchiya, S.: Single document summarization based on local topic identification and word frequency. In: Artificial Intelligence, 2008. MICAI'08. Seventh Mexican International Conference on. (2008) 37–41

[7] http://www.cs.kuleuven.be/ liir/projects/daisy/
[8] http://www.puppyir.eu

12. Wang, D., Zhu, S., Li, T., Gong, Y.: Multi-document summarization using sentence-based topic models. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, Suntec, Singapore, Association for Computational Linguistics (August 2009) 297–300
13. Dorr, B., Zajic, D., Schwartz, R.: Hedge Trimmer: a parse-and-trim approach to headline generation. In: Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5, Association for Computational Linguistics Morristown, NJ, USA (2003) 1–8
14. Marsi, E., Krahmer, E., Hendrickx, I., Daelemans, W.: Is sentence compression an NLG task? In: Proceedings of the 12th European Workshop on Natural Language Generation, Association for Computational Linguistics (2009) 25–32
15. Hori, C., Furui, S.: Speech summarization: an approach through word extraction and a method for evaluation. IEICE Transactions on Information and Systems **87** (2004) 15–25
16. Turner, J., Charniak, E.: Supervised and unsupervised learning for sentence compression. Ann Arbor **100** (2005)
17. Bouma, G., Van Noord, G., Malouf, R.: Alpino: Wide-coverage computational analysis of Dutch. In: Computational Linguistics in the Netherlands 2000. Selected Papers from the 11th CLIN Meeting. (2001)
18. Deschacht, K., Moens, M.F.: Semi-supervised semantic role labeling using the latent words language model. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)
19. Deschacht, K., Moens, M.F.: The Latent Words Language Model. In: Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning. (2009)
20. Martins, A., Smith, N., Xing, E.: Concise integer linear programming formulations for dependency parsing. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL09), Singapore. (2009)
21. Denis, P., Baldridge, J.: Joint determination of anaphoricity and coreference resolution using integer programming. In: Proceedings of NAACL HLT. (2007) 236–243
22. Roth, D., Yih, W.: Integer linear programming inference for conditional random fields. In: Proceedings of the 22nd international conference on Machine learning, ACM (2005) 743
23. Briscoe, T., Carroll, J., Watson, R.: The second release of the RASP system. In: Proceedings of the COLING/ACL. Volume 6. (2006)
24. Ordelman, R., de Jong, F., van Hessen, A., Hondorp, H.: Twnc: a multifaceted Dutch news corpus. ELRA Newsletter **12**(3/4) (2007) 4–7
25. Lin, C.: Rouge: A package for automatic evaluation of summaries. In: Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004). (2004) 25–26