# Chapter 19
# Question Answering of Informative Web Pages: How Summarisation Technology Helps

**Jan De Belder, Daniël de Kok, Gertjan van Noord, Fabrice Nauze, Leonoor van der Beek, and Marie-Francine Moens**

## 19.1 Introduction

The DAISY (Dutch lAnguage Investigation of Summarisation technology) project started from a practical problem. Many companies maintain a large website with informative content. The users of such a website (e.g., clients of the company, business partners) want to quickly find the information that is relevant for their information question without getting lost when navigating the company's website, and want immediately to be directed to the right part of information when typing an information need. Summarisation of the informative Web texts will help in finding the correct answer to the information need. Summarised and rhetorically classified segments of the Web page will help to automatically map a user's question with the relevant information on the page.

DAISY is joint work of teams of the Katholieke Universiteit Leuven, the Rijksuniversiteit Groningen and the company RightNow (formerly Q-go). The aim of DAISY is to develop and evaluate essential technology for automatic summarisation of Dutch informative texts. Innovative algorithms for Web page segmentation, rhetorical classification of page's segments, sentence compression and generation of well-formed Dutch text have been developed. In addition, a proof-of-concept demonstrator is being developed in collaboration with the company RightNow.

J. De Belder (✉) · M.-F. Moens
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001, Heverlee, Belgium
e-mail: jan.debelder@cs.kuleuven.be; Sien.Moens@cs.kuleuven.be

G. van Noord · D. de Kok
University of Groningen, Groningen, The Netherlands
e-mail: g.j.m.van.noord@rug.nl; d.j.a.de.kok@rug.nl

F. Nauze · L. van der Beek
RightNow, Amsterdam, The Netherlands
e-mail: fabrice.nauze@rightnow.com; leonoor.vanderbeek@rightnow.com

P. Spyns and J. Odijk (eds.), *Essential Speech and Language Technology for Dutch*, Theory and Applications of Natural Language Processing,
DOI 10.1007/978-3-642-30910-6_19, 

The remainder of this chapter is organised as follows. In the next section, we define the problem at hand. In Sect. 19.3, we discuss the cleaning and the segmentation of Web pages, which then can be used as input for further processing, such as by the rhetorical classifier, discussed in Sect. 19.4. Then, we continue with the sentence compression and sentence generation aspects of the project, discussed in Sect. 19.5 and 19.6 respectively. Finally, we discuss the demonstrator, to show the different methods, and end with the conclusion in Sect. 19.8.

## 19.2 Problem Definition

The general aim of the project is to develop and implement essential methods and supporting algorithms for summarisation of informative texts written in Dutch, and apply and evaluate them with texts in the financial and social security domain that are currently posted on the World Wide Web.

More specifically, the aim is to develop novel and robust technologies for (1) Segmentation and salience detection of content; (2) Single-sentence compression and sentence generation; (3) Rhetorical classification of informative text. For testing and evaluation purposes a demonstrator is being built that generates complementary types of summary information: (1) A headline type summary of a single text or text segment; (2) A short textual summary composed of compressed sentences; (3) Metadata that describes the rhetorical role (e.g., procedure, definition) of the text or text segment of which the summary is made.

For example, take the following text fragment:

```
''SNS Bank heeft maatregelen getroffen voor veilig Internet Bankieren''
 (SNS Bank has taken measures to perform bank transactions in a safe way).
```

In the context of the discourse, the sentence can be reduced to

```
''Maatregelen voor veilig Internet Bankieren''
(Measures to perform bank transactions in a safe way).
```

Also, detected rhetorical roles can be attached as meta-data to texts and their summaries. For example:

```
Example of a procedure ''Verzenden met EasyStamp'' (Send with EasyStamp)

''selecteer het adres of typ postcode en huisnummer in
kies het gewicht van het poststuk
selecteer een envelop of etiket (veel soorten en maten zijn al gedefinieerd)
kies eventueel voor een logo of afbeelding die u mee wilt printen
druk op de printknop''

(select the address or type postcode and house number
choose the weight of the mail piece
select an envelope or label (many types and sizes are defined)
choose optionally a logo or image that you want to print
push the print button)
```

In the example above, the fragment would be classified as a procedure, one of the six types of rhetorical roles we detect.

Essential in summarisation is the reduction of content to its most essential (salient) constituents and the generation of a concise summary text or other representation (e.g., in the form of concepts) that can be easily and efficiently processed by humans or by machines. Research into automated summarisation of text goes back several decades, but becomes increasingly important when information has to be selected from or sought in large repositories of texts. For an overview on text summarisation we refer to [11, 28], the proceedings of the yearly Document Understanding Conference (DUC) (2000–2007), and the proceedings of their successor, i.e. the Text Analysis Conferences (TAC) (2008–2012) Many current summarisation systems just extract sentences that contain content terms occurring frequently in the text, that occur at certain discourse positions, that contain certain cue terms (e.g., "in conclusion"), or learn the importance of these and other sentence scoring features from a training set of example texts and their summaries. Hence, the state of the art in summarisation is still far from truly abstractive summarisation, fusion of information from different texts, generalising content, and producing fluent, sensible abstracts. We see a current research interest in moving beyond extraction towards compressing and generating suitable summary sentences (e.g., [3, 8, 21, 31]). However, research into summarisation of Dutch texts is limited (e.g.,[24]: summarisation of court decisions; [32, 33]: summarisation of speech; [23]: summarisation of magazine articles). Studies that integrate into the summarisation certain pragmatic communication roles of the content are new. Segmentation and summarisation of informative texts that contain, for instance, instructions and procedural content are seldom researched.

A text may fulfill various pragmatic communication roles. For instance, it may describe a procedure, inform about a fact, or give a definition. Such roles are signaled by certain rhetorical linguistic cues. It is important to type a text (segment) according to its rhetorical function, as such typing has been proven a valuable part in summarising textual content [9, 30]. In this project, we use rhetorical typing in order to answer certain types of questions with text to which a suitable role is attached in a question answering system. Rhetorical structures of texts have been studied by Mann and Thompson [19] and used for summarisation of expository texts by Marcu [20].

This research extends previous work on text segmentation. After studying the corpora, and based on the literature of discourse theories, we defined a limited, but important set of rhetorical roles that are characteristic of the informative texts (e.g., definition, procedure, rule, . . .). These also correspond to the types of questions with which people interrogate the finance and social security texts.

In Fig. 19.1, we schematically represent the different components, and how they interact with each other.

## 19.3 Cleaning and Segmentation of Web Pages

A first step in analysing text on Web pages consists of extracting the text from the Web page. For humans this is a trivial task: a single glance at a page suffices to distinguish the main content from the less important information. However, when

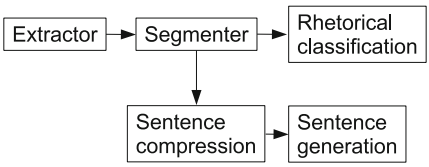


**Fig. 19.1** Overview of different components and their interaction

only looking at the HTML code, it is often difficult to determine exactly where the main content starts and ends. Header, footers, menus, advertisements, ..., these are all elements that have to be taken into account, and dealt with properly.

The segmentation of Web pages goes a step further. For many Information Retrieval tasks a simple bag-of-words representation is sufficient, but here we also want the structural layout of the text. This means segmenting the text into sections, subsections, paragraphs,... and attaching the correct sections titles.

### *19.3.1 Content Extraction*

The method we use for the extraction of the content performs only a very shallow analysis of the Web page. It does not depend on strong assumptions on the structure or content of the Web page and is fully language independent. The main idea behind the method is that a Web page has both content text and garbage text, but that the content texts tend to be continuous, long text with little structural markup, and that the garbage text tends to be short texts with a lot of structural markup. We make the following weak assumptions: The first assumption states that the text representing the content is separated from the garbage text with one or more markup tags. The second assumption states that no garbage text occurs in the main content, e.g. that the main content text is continuous (not taking into account the markup tags). The third and most important assumption states that the main content of the text contains less structural markup tags than the garbage text.

The method first locates a subset of markup tags that modify the structure of the Web page. These tags include, but are not limited to P, TABLE, BR, DIV, H1, H2 and LI tags. We ignore the tags that do not modify the structure of the Web page, such as B, A and FONT, and we also ignore data that is not content-related, such as JavaScripts, style definitions and HTML comments. We then transform the structured HTML page to a linear list of text strings $L = \{s_1, \ldots, s_n\}$. We parse the structure of the Web page using a robust HTML parser, that will, when presented with a not well-structured HTML page perform a best-effort parse. This parser visits every node in the HTML structure. If a node containing text is encountered, this text is added to the last text string in $L$. If a markup tag that modifies the structure of the Web page is encountered, $L$ is extended with one empty string. We continue this process until the entire Web page is parsed.

We build a graphical representation of the array $L$ in Fig. 19.2 where the x-axis represents the position of the array and the y-axis represents the length of the strings

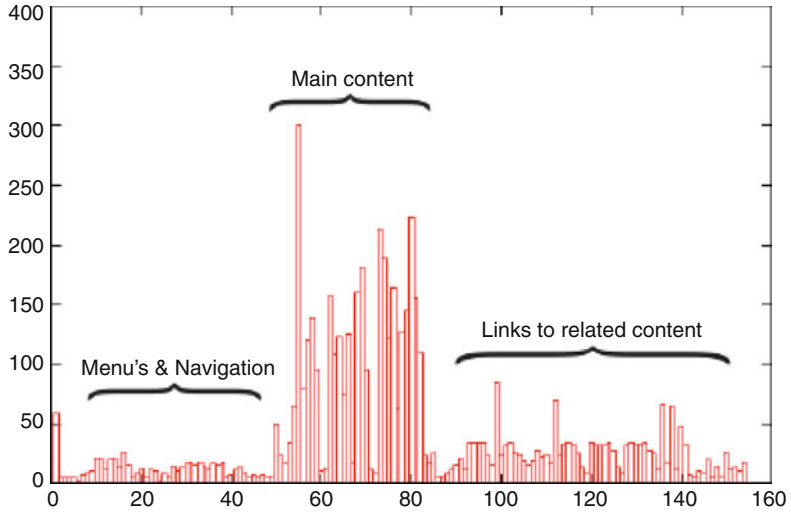


**Fig. 19.2** Example plot of the document density

at the different positions. In a second step we analyse this graph to find the main content in the Web page. Typically, the main content for a Web page containing news articles is located in the region of L that has the highest density. We therefore convert the problem of extracting the main content of a Web page to the problem of selecting the highest density region of L, for which we have designed a simple but effective algorithm.

Although the method as a whole is very simple, it incorporates several interesting ideas. First of all, it does not depend on the structure of any particular Web site, but uses a notion of document density which can be expected to be universal for most Web sites containing news articles. Secondly, it does not depend in any way on the text and is thus fully language independent. Thirdly, it relies only on a limited amount of the HTML-markup, thus making allowances for dirty and non-well structured Web pages. For more details including the evaluation of the method on benchmarking corpora, see [1].

### *19.3.2 Segmentation*

The nature of the Web pages in the corpora makes additional segmentation straightforward. HTML formatting tags present in the source files indicate text blocks (DIV), titles (H1, H2, H3), paragraphs (P), lists (UL, OL, DL), . . . providing strong clues about both the structure and the content (by looking at titles) of the text. An additional advantage is the generic nature of HTML, so the structure of any well-formed HTML page can be determined in a uniform fashion. Blocks of continuous text are further segmented into individual sentences.

The structure of the text provides cues for the rhetorical classification and the detection of the salient content in the document.

### *19.3.3 Corpora*

The Web pages used in most of the experiments in the remainder of this chapter, were provided by RightNow. The company contacted several of its clients for the use of the data. Among those we have KLM,[1] UWV,[2] and SVB.[3]

## 19.4 Rhetorical Classification

For the purpose of better distinguishing parts of the Web pages, we classify the sentences as having a rhetorical role. We made a distinction between six relevant high level roles:

- **Definition** (DEF) A definition of a term, explaining its meaning.
- **Reference** (REF) A reference to another source for more background information, or a different source with the answer to the question (e.g. a phone number).
- **Information** (INF) An explanation or clarification about something (who, where, when, how much, which, why, ...-questions). In contrast to the commanding undertone that is present in Rules, there is a softer informative undertone that offers possibilities ('you can'), and not obligations ('you have to').
- **Meta-Information** (MIN) An explanation of why, which, and to what end the information is given on the page. It is information about the information that can be found on the page.
- **Procedure** (PRO) How a certain process is executed, or the different steps that need to be taken in order to complete something.
- **Rule** (RUL) A way one has to behave, i.e. an appointed or accepted norm, obligation, right or expectation, given in a commanding voice ('you have to', instead of 'you can').

These roles were developed by inspecting the corpora. They can be broken down further, e.g. a rule can be subdivided in a precondition and a postcondition. However, this could pose problems, as multiple of these more fine-grained roles can occur in a single sentence, which makes the classification task more difficult, and the training data will become sparse for some of the classes. We believe that these six high level roles are sufficient for practical use, and focus on them for now.

[1] http://www.klm.com/, the Royal Dutch Airlines

[2] http://www.uwv.nl/

[3] http://www.svb.nl/

### *19.4.1 Experiments*

A set of 374 documents, selected randomly from the different corpora, was annotated with these roles. In total, we have 9,083 labelled sentences. The largest classes are "information" (32.8 %), "rule" (29.0 %) and "reference" (24.4 %). The other classes are smaller (8.8 % contains a "procedure", 3.5 % is a "definition", and 1.6 % is "meta-information"). It was however not always clear to which role a statement belonged. For instance, the difference between a piece of information and a rule are subject to personal judgement.

#### 19.4.1.1 Baseline

As a baseline, we started by treating the problem as a multiclass text classification problem. We use unigrams and bigrams to represent the lexical aspect, and unigrams, bigrams, and trigrams of the Part-Of-Speech tags. We also include the POS tag of the first word as a feature, and several binary indicators for the presence of an imperative verb, a Multi-Word Unit, a wh-word, an auxiliary verb, possessives, and colons. The positional properties inform about the position in the paragraph, the depth in the hierarchy, and whether the sentence is actually a title. Finally, we include some statistics such as the number of words, the number of punctuation marks, and the average number of characters per word. We only kept the most significant features, according to a $\chi^2$ test at $p < 0.05$.

We experimented with several algorithms, and found multinomial naive Bayes to be favourable compared to a maximum entropy classifier and a support vector machine. The results (of a ten-fold cross validation) show an accuracy of up to 70 % and a macro-averaged F1-measure up to 52.54 %. The rest of the results can be found in Table 19.1.

#### 19.4.1.2 Improved Algorithm

Having exhausted the possible features and classification algorithms, we made use of additional information to improve results. Since the role of a sentence is dependent on the role of its surrounding sentences, and its position in the hierarchy, we try to find a globally optimal assignment for all the sentences in a document. We do so by building simple transition models, where we assign a probability of a label based on the label of the previous sentences, or the label of the sentence that is the parent in the hierarchy. Combining this with the probabilistic output of the multinomial naive Bayes classifier, we can find an assignment for all the sentences that maximises the probability of the document as a whole, by solving the corresponding optimisation problem with an Integer Linear Programming formulation.

**Table 19.1** $F_1$ scores of the different methods. The column labelled *Baseline* indicates the baseline method, after applying feature selection. The *second column* indicates the sequential method, and the *third* the hierarchical method. The *last column* combines the two latter methods

| Class | Baseline | Sequential | Hierarchic | Sequential + Hierarchic |
|---|---|---|---|---|
| DEF | 46.39 % | 55.81 % | 54.4 % | 58.06 % |
| DVW | 82.78 % | 84.03 % | 83.34 % | 84.4 % |
| INF | 60.82 % | 62.98 % | 62.38 % | 64.31 % |
| MIN | 39.55 % | 43.84 % | 39.81 % | 42.27 % |
| PRO | 34.68 % | 39.76 % | 36.07 % | 43.11 % |
| REG | 51.0 % | 53.81 % | 51.57 % | 53.69 % |
| Accuracy | 70.73 % | 74.06 % | 72.55 % | 75.21 % |

By using the additional information given by the segmenter, and finding a globally optimal solution, we have obtained an average accuracy of 75 %, and a macro-averaged 57.64 % F1 score, thereby improving the baseline accuracy with 5 %. The complete results can be found in Table 19.1.

### *19.4.2 Conclusions and Future Work*

In this component we have looked at assigning a rhetorical role to sentences in an informative document. This is a novel task, and there is no previous work with which we can compare. We initially treated the problem as a text classification problem. In order to improve the results, we combined this basic classifier with information from the previous component, i.e. the segmentation. Now a globally optimal assignment is found, and this led to improved results.

The obtained results are probably also the upper limit that can be reached without annotating more data. The rhetorical classification is a difficult task, as often it is hard to distinguish between the different roles.

Another possible line of research, is by using more data in an unsupervised setting. E.g. by taking the first sentences of each Wikipedia article, it is straightforward to obtain a corpus consisting of definitions. These can then be used to train a better classifier for definitions. A similar approach can be followed for procedures, e.g. by retrieving a set of instructional texts.

## 19.5 Sentence Compression

There exist a myriad of methods for the task of sentence compression, but the majority of these are hard to use in this case. The majority of methods learn how to compress sentences by learning from training data [13, 21, 31]. However,

manually creating training data is a time consuming and expensive task. Moreover, the few corpora that are available for Dutch, are from a completely different domain. Another aspect of this project that in a way limits the range of possibilities for sentence compression algorithms, is that Dutch is not a context free language, which means that we can't make use of the large number of methods for English that build on a Probabilistic Context Free Grammar (e.g. [8, 13]). Therefore, in this research we focused our attention on unsupervised methods, that are not too dependent on the output format of the parser.

We view sentence compression in a word removal setting. An advantage of such an approach is that sentence compression can be seen as a machine learning task, and many methods can be applied to it. In the literature we find, among others, a noisy channel model approach ([8, 13, 31]), a method based on integer linear programming [4] and a large margin online learning approach [21].

In this section we will define a uniform framework for compressing sentences using language models and optimisation. At the core of the algorithms lies the following problem: choose a subset of the words that maximise the probability of the compressed sentence in a language model. The major difference between the methods is the type of language model that is being used. Choosing this optimal subset of words can be done by solving an Integer Linear Programming problem. Below we sketch the broad ideas behind the methods.

### *19.5.1 Constrained Language Model Optimisation*

We investigated three unsupervised methods, which we modelled in a similar fashion. Each of the methods share a similar problem formulation. They start from binary variables $a_i$ for each word $w_i$ in the sentence to be compressed. These variables can only have a value of 1 or 0, the former indicating that the word is included in the sentence, the latter indicating that $w_i$ is not included in the compressed sentence.

With these $a_i$ variables, and other variables depending on the model, we create a linear objective function. An optimal solution for the sentence compression problem is then found by finding an assignment for the variables that maximises the objective function. The difference between the methods lies in how they fill in this objective function.

#### 19.5.1.1 Optimising an n-Gram Language Model

For a bigram language model, this roughly translates to assigning values of 0 or 1 to variables $x_{ij}$, each $x_{ij}$ meaning that the bigram $w_i\ w_j$ is present in the compressed

sentence.[4] An optimal solution is then found by maximising:

$$\sum_{i=0} \sum_{j=i+1} x_{ij} P(w_j|w_i) \tag{19.1}$$

with $P(w_j|w_i)$ the probability that the word $w_i$ is followed by the word $w_j$,

To ensure that the output is grammatically correct, and doesn't lose semantic information, an additional set of rules is applied, that are enforced in the form of constraints. These are based on a syntactic analysis of the sentence. The constraints state for example that when a verb is included in the compressed sentence, its subject and object also have to be included, and that a negation can not simply be removed from the word it modifies, etc.

#### 19.5.1.2 Optimising a Dependency Language Model

A disadvantage of the previous method, is that the n-gram language model only finds fluent sentences locally. By using a language model defined over dependency trees, such as in [7], this problem is alleviated. In a dependency tree representation, words that are syntactically close together, are also close together in the model.

For each dependency ending in word $a_i$ we have the parent word $h_i$, and $l_i$, the label of the dependency relation between $a_i$ and $h_i$. The goal is then to maximise the following equation:

$$\sum_{i} a_i P(l_i|h_i) \tag{19.2}$$

with $P(l|h)$ the probability that a certain word $h$ has a child with the dependency label $l$. This latter is estimated as $P(l|h) = \frac{Count(h,l)}{Count(h)}$, where the counts are obtained by parsing a sufficiently large corpus. E.g. most verbs have a subject, so $P(subj|have)$ will be high.

#### 19.5.1.3 Optimising an Unfactored Dependency Language Model

A disadvantage of the method in [7], is that the probability of the children of a word are estimated separately ($P(l|h)$, the probability of word $h$ having a child with label $l$ between them). Our parsed corpus is however large enough, so that we can estimate $P(l_1, l_2, ..|h)$: the probability of a word having a set of children (e.g. the probability of a verb having a subject *and* an object, instead of the individual probabilities.

[4] In practice we use a trigram model, but for simplicity this is left out.

Additional Constraints

One of the most important functions of the constraints is to ensure that the problem is solved correctly. E.g. in equation 19.1, $x_{13}$ can only have value 1 if $a_1 = 1, a_2 = 0, a_3 = 1$. Other possibilities with these constraints are stating that $\sum_{i=1}^{n} a_i \geq lowerbound$, to specify a minimum number of words.

Significance Model

To ensure that the compressed sentence contains the most important information, we modify the objective function, so that an additional 'bonus' is given for including a specific word. For each word, we calculate the importance with the following equation:

$$I(w_i) = \frac{l}{N} f_i log \frac{F_a}{F_i} \tag{19.3}$$

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

### *19.5.2 Evaluation and Results*

Using current evaluation measures, we can show that our unsupervised methods perform comparably with supervised methods. We not only evaluated on our own annotated small subset of the corpus, the results of which are available in Table 19.2, but also on existing corpora for sentence compression, of which our findings are in preparation. From Table 19.2, we can see the difference between the methods. Using only the n-gram language model and grammaticality constraints, the output is not so grammatical, but contains the most important information. When using language models based on the dependency trees, the output becomes more grammatical, but the score for the importance goes down, despite the longer sentences. The difference lies in the fact that the last two methods don't take into account the lexical items in the leaves of the dependency tree.

We also correlated different automatic evaluation measures with human judgement. Our results show that for Dutch, the evaluation measure based on the parse tree is the most correlated. This measure also takes the grammaticality into account, because if a sentence is ungrammatical, the parser will not be able to capture the dependencies between the words.

The annotation process of the informative texts was very enlightening. Annotators found it very difficult to compress sentences without the proper context. When

**Table 19.2** Human ratings for each of the three methods, on a five point scale (5 being the highest score, 1 the lowest), grading the grammaticality and importance aspect. The *last column* indicates the average number of words in the compressed sentence

| Method | Grammaticality | Importance | AvgNbWords |
|---|---|---|---|
| n-gram LM | 2.60 | 3.23 | 12.1 |
| Dependency LM | 3.28 | 2.95 | 12.7 |
| Joint dependency LM | 3.67 | 2.62 | 12.9 |

faced with the complete text, this posed less of a problem, although it was still harder in comparison to texts containing a lot of redundant information.

In a practical setting, it is often faster to use a method with a language model based on dependency trees, rather than one with an n-gram language model. The disadvantage is that this yields a lower importance score, but this can be alleviated by using the Significance model. The trade-off between the two models then has to be estimated on a small validation set.

We refer the interested reader to other publications for more information [6].

## 19.6 Sentence Generation

### *19.6.1 Introduction*

Since the sentence compression component deletes words, it is possible that the word order has to be changed. In order to reorganise the ordering of the words, we use a sentence realiser that, given the dependencies required in a sentence, arranges them for a fluent result.

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

Alpino [25] is a parser for Dutch which includes an attribute-value grammar inspired by HPSG, a large lexicon, and a maximum entropy disambiguation component. Dependency structures are constructed by the grammar as the value of a dedicated attribute. These dependency structures constitute the output of the parser.

In generation, the grammar is used in the opposite direction: we start with a dependency structure, and use the grammar to construct one or more sentences which realise this dependency structure. Dependency structures that we use in generation contain less information than the dependency structures that are the output of parsing. For instance, information about word adjacency, separable particles and punctuation are removed. The user can also decide to underspecify

certain lexical information. We call such dependency structures *abstract dependency structures* [16].

In the general case, a given dependency structure can be realised by more than a single sentence. For instance, the sentence *Na de verkiezingen beklijfden de adviezen echter niet* (*After the elections the advises did, however, not persist.*) is mapped to a dependency structure which can also be realised by variants such as *Na de verkiezingen beklijfden de adviezen niet echter*, or *echter beklijfden na de verkiezingen de adviezen niet*. Therefore, a maximum entropy fluency ranker is part of the generator. The fluency ranker selects the most appropriate, 'fluent', sentence for a given dependency structure.

### *19.6.2 Chart Generation*

In the Alpino generator, we use chart generation [12, 29]. This algorithm closely resembles bottom-up chart parsing, however guidance is provided by semantics rather than word adjacency.

For details of our sentence realiser, we refer to [16]. However, one interesting aspect of our realiser is that it implements top-down guidance differently than in previous work that we know of. Since the Alpino grammar is semantically monotonous [29], we could use a semantic filter that constrains generation. Such a filter excludes derivations where the semantics of the derivation do not subsume a part of the goal semantics. In our system, we use an even stronger approach: we instantiate each lexical item that represents a head in the dependency structure with its expected dependency structure. In this manner, it is not possible to construct partial derivations with dependency structures that do not subsume a part of the input dependency structure.

### *19.6.3 Fluency Ranking*

A sentence realiser can often produce many different sentences for a given input. Although these sentences are normally grammatical, they are not all equally fluent. We developed a fluency ranker that attempts to select the most fluent sentence from a list of sentences.

Different statistical models have been proposed for fluency ranking in the past, such as n-gram language models, maximum entropy models, and support vector machines [34]. As [34] shows, maximum entropy models perform comparably to support vector machines for fluency ranking, while having a shorter training time. For this reason, we use a conditional maximum entropy model in our fluency ranker.

In our model probability of a realisation $r$ given the dependency structure $d$ is defined as:

**Table 19.3** General Text Matcher scores for fluency ranking using various models

| Model | GTM |
|---|---|
| Random | 55.72 |
| Trigram | 67.66 |
| Fluency | 71.90 |

$$p(r|d) = \frac{1}{Z(d)} exp \sum_i \lambda_i f_i(d, r) \tag{19.4}$$

Where $f_i(f, r)$ is the value of feature $f_i$ in the realisation $r$ of $d$, $\lambda_i$ the weight of that feature, and $Z(d)$ normalises over all realisations of the dependency structure $d$. Training the model gives a set of feature weights $\Lambda$ that predicts the training data, but has as few other assumptions as possible.

Features are automatically extracted from the training data using feature templates. Our fluency ranker works with the following classes of features:

- *Word adjacency* is modelled using trigram language models of words and part-of-speech tags.
- *Shallow syntactic* features record rule applications and combinations of rule applications.
- *Syntactic* features describe various syntactic aspects of a realisation, such as fronting, depth and parallelism in conjunctions, and orderings in the middle-field.

### *19.6.4 Evaluation and Results*

To evaluate the fluency ranker, we first trained a fluency ranking model using the *cdbl* part of the Eindhoven corpus[5] (7,154 sentences). Syntactic annotations are available from the Alpino Treebank[6] [2].

We then evaluated this model using a part of the Trouw newspaper of 2001 from the Twente Nieuwscorpus. [7] Syntactic annotations are part of Lassy [8] [26], part WR-P-P-H (2,267 sentences). For each pair of a sentence and dependency structure in the treebank, we consider the sentence to be the gold standard, and use the dependency structure as the input to the generator. We then use the General Text Matcher method [22] to compute the similarity of the most fluent realisation and the gold standard sentence.

Table 19.3 compares random selection, a word trigram model, and our fluency ranking model. As we can see in this table, our maximum entropy fluency ranking model outperforms both the random selection baseline and the word trigram model.

[5] http://www.inl.nl/corpora/eindhoven-corpus

[6] http://www.let.rug.nl/~vannoord/trees/

[7] http://hmi.ewi.utwente.nl/TwNC

[8] http://www.inl.nl/corpora/lassy-corpus

### *19.6.5 Further Research*

In [14] we have compared various feature selection methods to reduce the size of the fluency ranking model and to get more insight into the discriminative features. We also developed the Reversible Stochastic Attribute-Value Grammar (RSAVG) formalism, that uses one model for both parse disambiguation and fluency ranking [17]. Subsequently, we have RSAVG to be truly reversible [15].

## 19.7 Proof-of-Concept Demonstrator

The developed technology is made publicly available through the demonstrator. This demonstrator is a Web-based interface that allows users to summarise sample texts, uploaded documents, or shorts texts, which the user enters in a textbox. A screenshot of the interface is shown in Fig. 19.3. For testing and evaluation purposes the demonstrator generates three complementary types of summary information: (1) A headline type summary of a single text or text segment; (2) A short textual summary composed of compressed sentences; (3) Metadata that describes the rhetorical role (e.g., procedure, definition) of the text or text segment of which the summary is made. The combination of the summaries and the metadata discriminate a text in a document base by the description of topics and the role of the text (segment) in the discourse.

Two lines of evaluation of the demonstrator will be pursued: an intrinsic and an extrinsic one. With intrinsic evaluation, the system's output is compared with humans' output and their congruence is computed. Extrinsic evaluation on the other hand, measures the quality as needed for other information tasks (e.g., filtering and retrieval).

We have performed an intrinsic evaluation with some common metrics from the Document Understanding Conference, namely 'Pyramid' [10] and 'Rouge' [18] and. When evaluating the demonstrator, the system output is compared against hand-made abstracts of the documents. Because of the problem of subjectivity of human summarisation, wherever possible three or more model summaries of the same text were collected. It is expected that good system-made summaries have a sufficient amount of congruence with at least one of the human-made summaries. The model summaries have been created by the company RightNow. Very often variant summaries made by different persons are available. In each step, both a baseline summary and the summaries generated by the demonstrator were compared with the model summary.

The effect of adding system-generated headline abstracts on retrieval will be measured. The summaries are used to assist the question answering system developed by RightNow in the search for precise answers to information queries posed by end-users. This extrinsic evaluation is very important. RightNow monitors the recall and precision of its question answering system. This data can be reused in

order to test whether recall and precision of the retrieval can be improved by adding automatically generated summaries to the system, or by replacing the hand-made abstracts with system summaries.

Currently, RightNow processes user questions based upon a lexical, syntactic and semantic analysis, which results in a formal representation. The application matches such representations against similar representations in a database. These database entries are the result of the linguistic analysis of "template questions". The template questions are created manually, and each question is associated with an answer, which may be a piece of content on the customer website, or a brief textual answer and a link to the relevant Web page.

We have manually crafted template questions and the short textual answers as one or more summarisations reflecting the gist of the target document, which is why we think that an applied summarisation system can replace or at least help a large part of the editorial procedure needed in the current setup. Furthermore, we hope to improve the retrieval by associating automatically created summaries to templates as an alternative for matching.

The obtained pyramid and rouge results of the DAISY summaries are comparable with what we see in the state-of-the-art literature of the DUC [27] and TAC [5] competitions organised by the National Institute of Standards and Technology (NIST) for the English language (where other types of texts such as news stories were summarised). Compared to uncompressed HTML text of the Web pages, there are few matching LCS (lowest common sub-sequences). This is mainly caused by three factors:

1. Even though the knowledge base content is linked to external Webpages, the match questions in the database try to model the way end-users formulate questions about the web content, it is not a model of the content itself.
2. The segmentation of the html text does not handle links and lists correctly.
3. The compression used for the evaluation is quite aggressive which has a great impact on matching sub-sequences.

This result was to be expected as content on informative Web pages is always important for a certain user and summarisation or compression is not always the correct answer to improve the matching in a question answering task.

We also evaluated whether summaries can replace match questions within the Intent Guide (IG) without loss in quality and whether the summaries can improve the current quality of the IG implementation. We run two experiments. In the first one only summaries were used for matching, in the second summaries were added to the model questions as match questions (the customer would therefore still need to create model questions – the questions displayed to the end-user as answer to his/her query).

The results of the first run show a match percentage of 30 % which is too low to replace the IG match question. However we do get new matches with the second test set which is positive. The second run shows that the addition of model questions improves the results greatly and that the summaries might be used as extra matching questions (improving the system but invisible to the end-user).

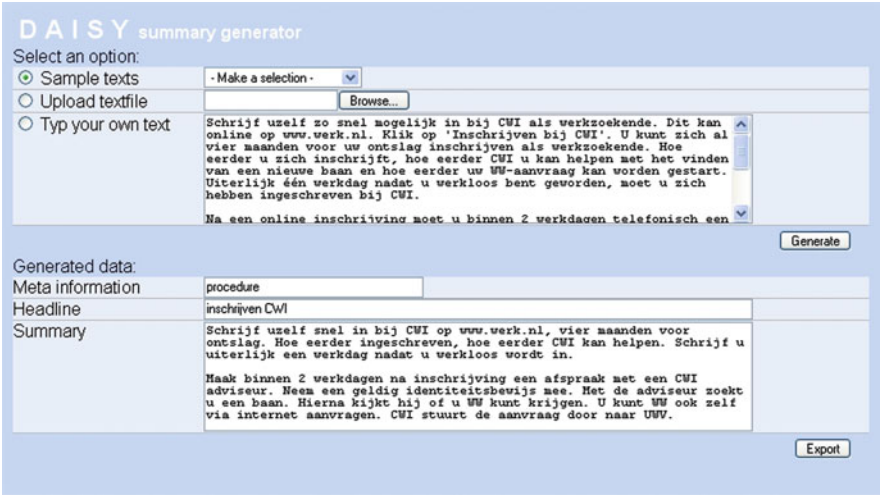


**Fig. 19.3** Demonstrator interface

## 19.8 Conclusions

The novelty of our approach lies in (1) Classification of the rhetorical role of a text segment or sentence, using text automatically extracted from Dutch informative Web pages; (2) Improvements of current sentence compression technologies for Dutch texts; (3) Development of standard text generation technology for Dutch – integrated with the standard Dutch text analysis tools.

These tasks regard essential tasks in summarisation of informative content. The summarisation demonstrator can already be considered as an application. Because in informative Web pages any content is important in a certain circumstance for some user, it is difficult to compress this content. But, DAISY has contributed to generating additional paraphrases to the ones already used by RightNow for matching questions and answers.



## References

1. Arias, J., Deschacht, K., Moens, M.F.: Content extraction from multilingual web pages. In: Proceedings of the 9th Dutch-Belgium Information Retrieval Workshop. University of Twente, Enschede, The Netherlands (2009)
2. van der Beek, L., Bouma, G., Malouf, R., van Noord, G.: The Alpino dependency treebank. In: Computational Linguistics in the Netherlands (CLIN), Groningen, The Netherlands (2002)

3. Clarke, J., Lapata, M.: Models for sentence compression: a comparison across domains, training requirements and evaluation measures. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, pp. 377–384. Association for Computational Linguistics, Sydney, Australia (2006)
4. Clarke, J., Lapata, M.: Global inference for sentence compression: an integer linear programming approach. J. Artif. Intell. Res. **31**(1), 399–429 (2008)
5. Dang, H.T., Owczarzak, K.: Overview of the TAC 2009 summarization track. In: Proceedings of the Second Text Analysis Conference (TAC2009), Gaithersburg, Maryland, USA (2009)
6. De Belder, J., Moens, M.F.: Integer linear programming for Dutch sentence compression. In: Computational Linguistics and Intelligent Text Processing, Iaşi, Romania, pp. 711–723 (2010)
7. Filippova, K., Strube, M.: Dependency tree based sentence compression. In: Proceedings of the Fifth International Natural Language Generation Conference, pp. 25–32. Association for Computational Linguistics, Columbus, Ohio, USA (2008)
8. Galley, M., McKeown, K.: Lexicalized Markov grammars for sentence compression. In: The Proceedings of NAACL/HLT, Rochester, NY, USA, pp. 180–187 (2007)
9. Hachey, B., Grover, C.: Automatic legal text summarisation: experiments with summary structuring. In: Proceedings of the 10th International Conference on Artificial intelligence and Law, pp. 75–84. ACM, Bologna, Italy (2005)
10. Harnly, A., Nenkova, A., Passonneau, R., Rambow, O.: Automation of summary evaluation by the Pyramid method. In: Proceedings of the Conference of Recent Advances in Natural Language Processing (RANLP), Borovets, Bulgaria (2005)
11. Hovy, E., Marcu, D.: Automated text summarization. In: The Oxford Handbook of Computational Linguistics pp. 583–598. Oxford University Press (2005)
12. Kay, M.: Chart generation. In: Proceedings of the 34th Annual Meeting on ACL, pp. 200–204. ACL, Santa Cruz, California, USA (1996)
13. Knight, K., Marcu, D.: Statistics-based summarization-step one: sentence compression. In: Proceedings of the National Conference on Artificial Intelligence, pp. 703–710. MIT, Austin, Texas (2000)
14. de Kok, D.: Feature selection for fluency ranking. In: Proceedings of the 6th International Natural Language Generation Conference, pp. 155–163. Association for Computational Linguistics, Trim, Co. Meath, Ireland (2010)
15. de Kok, D.: Discriminative features in reversible stochastic attribute-value grammars. In: Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop, pp. 54–63. Association for Computational Linguistics, Edinburgh (2011). http://www.aclweb.org/anthology/W11-2708
16. de Kok, D., van Noord, G.: A sentence generator for Dutch. In: Proceedings of the 20th Computational Linguistics in the Netherlands conference (CLIN), Utrecht, The Netherlands (2010)
17. de Kok, D., Plank, B., van Noord, G.: Reversible stochastic attribute-value grammars. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, pp. 194–199. ACL, Portland, Oregon, USA (2011)
18. Lin, C.Y.: ROUGE: a package for automatic evaluation of summaries. In: Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Barcelona, Spain (2004)
19. Mann, W., Thompson, S.: Rhetorical structure theory: toward a functional theory of text organization. Text-Interdiscip. J. Study Discourse **8**(3), 243–281 (1988)
20. Marcu, D.: The theory and practice of discourse parsing and summarization. The MIT Press, Cambridge (2000)
21. McDonald, R.: Discriminative sentence compression with soft syntactic evidence. In: Proceedings of EACL, vol. 6, Trento, Italy, pp. 297–304 (2006)
22. Melamed, I.D., Green, R., Turian, J.: Precision and recall of machine translation. In: HLT-NAACL, Edmonton, Canada (2003)

23. Moens, M.F., Angheluta, R., Dumortier, J.: Generic technologies for single-and multi-document summarization. Inf. Process. Manag. **41**(3), 569–586 (2005)
24. Moens, M.F., Uyttendaele, C., Dumortier, J.: Abstracting of legal cases: the salomon experience. In: Proceedings of the 6th international conference on Artificial Intelligence and Law, pp. 114–122. ACM, Melbourne, Victoria, Australia (1997)
25. van Noord, G.: At last parsing is now operational. In: TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues Naturelles, pp. 20–42. Leuven, Belgium (2006)
26. van Noord, G., Schuurman, I., Bouma, G.: Lassy syntactische annotatie, revision 19053 (2010)
27. Paul, O., James, Y.: An introduction to duc-2004. In: Proceedings of the 4th Document Understanding Conference (DUC 2004), Boston, MA, USA (2004)
28. Radev, D., Hovy, E., McKeown, K.: Introduction to the special issue on summarization. Comput. linguist. **28**(4), 399–408 (2002)
29. Shieber, S.: A uniform architecture for parsing and generation. In: Proceedings of the 12th COLING conference, Budapest (1988)
30. Teufel, S., Moens, M.: Summarizing scientific articles: experiments with relevance and rhetorical status. Comput. Linguist. **28**(4), 409–445 (2002)
31. Turner, J., Charniak, E.: Supervised and unsupervised learning for sentence compression. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 290–297. Association for Computational Linguistics, Ann Arbor, USA (2005)
32. Vandeghinste, V., Pan, Y.: Sentence compression for automated subtitling: a hybrid approach. In: Proceedings of the ACL Workshop on Text Summarization, Barcelona, Spain, pp. 89–95 (2004)
33. Vandeghinste, V., Tjong Kim Sang, E.: Using a parallel transcript/subtitle corpus for sentence compression. In: Proceedings of LREC 2004. Citeseer, Lisbon, Portugal (2004)
34. Velldal, E.: Empirical realization ranking. Ph.D. thesis, University of Oslo, Department of Informatics (2008)