Text-to-text generation for question answering

Wauter Bosma, Erwin Marsi, Emiel Krahmer and Mariët Theune

Abstract When answering questions, major challenges are (a) to carefully determine the content of the answer and (b) phrase it in a proper way. In IMIX, we focus on two text-to-text generation techniques to accomplish this: content selection and sentence fusion. Using content selection, we can extend answers to an arbitrary length, providing not just a direct answer but also related information so to better address the user's information need. In this process, we use a graph-based model to generate coherent answers. We then apply sentence fusion to combine partial answers from different sources into a single more complete answer, at the same time avoiding redundancy. The fusion process involves syntactic parsing, tree alignment and surface string generation.

1 Introduction

Answering specific types of trivia style (so-called 'factoid') questions is often taken as the core domain of *question answering* (QA) research. An example of such a question would be: *what is RSI?* But what is the correct answer to such a question? Ostensibly, this is a definition question, and a plausible answer is something like *RSI means Repetitive Strain Injury*. But will this answer the need for information of the person asking the question? In general, it seems that even if an unambiguous

Wauter Bosma
VU University Amsterdam, e-mail: w.bosma@let.vu.nl

Erwin Marsi
Norwegian University of Science and Technology, e-mail: emarsi@idi.ntnu.no

Emiel Krahmer
Tilburg University, e-mail: e.j.krahmer@uvt.nl

Mariët Theune
University of Twente, e-mail: m.theune@ewi.utwente.nl

question is posed, users appreciate more information than a direct answer (Strzalkowski et al., 2000). Someone querying a system about RSI may be interested to know what the abbreviation stands for, but may also like to know what it actually *is*. Bates (1990) helps explaining the findings of Strzalkowski et al. by viewing an information search as a 'berry picking' process. Consulting an information system is only part of a user's attempt to fulfill an information need. It's not the end point, but just one step whose result may motivate a follow-up step. The 'factoid answer approach' fails to show leads to related information, which may trigger follow-up questions. Bakshi et al. (2003) show that when answering questions, increasing the amount of text returned to users significantly reduces their number of queries, suggesting that users utilize related information from the text surrounding the answer.

In short, the raw response of a question answering system is often not a suitable answer. Text-to-text generation can help transforming a QA response into an appropriate answer. Generating an answer involves two core decisions to be made: (1) which information should be included, and (2) how that information should be presented. Decision (1) requires *content selection*, which is the process of finding the boundary between useful information and superfluous information. Decision (2) involves choosing an optimal formulation.

In this chapter, we describe our efforts in text-to-text generation within the IMOGEN project. In particular, we describe two focus areas of research to improve the quality of the answer: (a) graph-based content selection to improve the answer in terms of usefulness, and (b) sentence fusion to improve the answer in terms of formulation. We use sentence fusion to join together multiple sentences in order to eliminate overlapping parts, thereby reducing redundancy. The results of this work have been applied in the IMIX system. This system uses a question answering system to pinpoint fragments of text which are relevant to the information need expressed by the user. A content selection system then uses these fragments as entry points in the text to formulate a more complete answer. Sentence fusion is applied to manipulate the result in order to increase the fluency of the text.

For example, for the question '*what is RSI*', the content selection system may find several passages which may be used in the answer:

1A RSI means *repetitive strain injury*.
1B Repetitive Strain Injury is generally caused by a mixture of poor ergonomics, stress and poor posture.
1C Repetitive Strain Injury may occur for instance in people who regularly work with a display device.

The system could return an answer by just enumerating the above sentences, but this answer would contain some degree of redundancy because the term *repetitive strain injury* occurs in each of the sentences. By using sentence fusion, we can fuse the last two sentences and generate a more fluent answer with less redundancy:

RSI means *repetitive strain injury*. Repetitive Strain Injury is generally caused by a mixture of poor ergonomics, stress and poor posture, and may occur for instance in people who regularly work with a display device.

In this chapter, we present the progress we made in developing text-to-text generation techniques for question answering. The next two sections – section 2 and 3 – are dedicated to content selection and sentence fusion respectively. The chapter closes with some final words and an outlook in section 4.

2 Graph-based content selection

Not boring anyone with irrelevant details and, at the same time, not withholding the essentials. That is the essence of content selection. It can also be seen as an optimization issue: more information takes the user more time to process, while less information may increase the number of interactions. The greatest efficiency is achieved when exactly that is said which is relevant to the user.

In this section, we describe a framework for content selection which is based on the notion of *contextual salience* – all evidence of salience of a particular content unit is based on the salience of related content units (its context). The underlying data model is based on graph theory – a paradigm which is excellently equipped for representing relations between content units, thus modelling coherence and redundancy in text.

Our model separates the actual content selection from the detection of coherence and redundancy relations. The relation detection process results in a graph, and the content selection algorithm uses that graph to select the sentences to include in the summary. This separation makes it possible to replace the relation detection algorithms or the content selection algorithm without changing anything else. It is also possible to combine several relation graphs and use the combined graph for content selection.

As such, our graph-based model does not prescribe the nature of the relations or algorithms used to find the relations: the relations may represent coherence relations, co-reference relations, lexical chains, cosine similarity, or any other paradigm that may model (semantic) relations in text and that is compatible with the graph-based representation. Each of those methods may be individually implemented and evaluated, possibly in combination with other methods. We evaluate the model using sentences as content units and cosine similarity as a feature to find relations between sentences.

2.1 Related work

Content selection – choosing what to include and what not to include in a summary – is useful in answer presentation; it is also a subtask of summarization. A typical text summarization system transforms a source document (or a set of documents) into a more concise document by: (1) splitting up the source into individual *content units*, (2) selecting the most salient content units, and (3) composing a summary of those content units. The system may exploit contextual information such as a user question. Because of the availability of resources and tools in automatic summarization, we evaluate our algorithms for content selection in the context of a summarization system.

Content units may vary in granularity from paragraphs to phrases. Often, sentences are chosen as content units because they are reasonably fine-grained information units and, at the same time, the possibility of ungrammatical results can be avoided (which is a major challenge in phrase-based summarization).

Rather than viewing a summary as a single text, summarization systems typically perform content selection by determining the relevance of each passage independently, and then composing a summary of the top ranking passages. Classical features for scoring sentences include the presence of cue phrases, term frequency, stop word lists, etc. (Edmundson, 1969; Luhn, 1958). Assessing the relevance of each sentence individually, these systems neglect the internal structure of the summary, despite insights in discourse organization which claim that meaning is tightly related to discourse organization (e.g. Mann and Thompson, 1988). The meaning in a text is not merely the sum of the meaning in its passages, but a passage should be interpreted in the context shaped by other passages. For example, given the two passages below, the second passage would have little meaning if the context provided by the first were omitted. Hence, a generic summarization system should include the second sentence in a summary only if the first is also included.

2A A commercial airliner crashed in northwestern Iran on Wednesday.

2B All 168 people on board were killed.

If content selection is to result in an answer to a user query or question, this makes the need for dealing with coherence even more pressing. While a generic summary (i.e., a summary which is generated without user input) should be internally coherent, a query-based summary should also be coherent with respect to the query. Similarly, the task of multi-document summarization (if the answer is drawn from multiple documents) introduces the need to deal with redundancy, as a summary should not mention the same thing twice.

A number of ad-hoc solutions to dealing with redundancy and coherence emerged in response to the challenges of multi-document and query-based summarization. For instance, Carbonell and Goldstein (1998) introduced the concept of *marginal relevance* to handle redundancy. They build up a summary by adding sentences one by one, with a bias toward sentences which contain new information with respect to already-selected sentences.

Barzilay and Elhadad (1997) modeled coherence by dividing the source into topics by identifying *lexical chains*. They composed summaries of one sentence from each of the strongest topics, as to maximize coverage. The summarization system of Blair-Goldensohn and McKeown (2006) prioritizes sentences in the summary which have a coherence relation to another summary sentence. Each of these answers to the problem of coherence represents a minor change to an existing summarization system, rather than an integral model of coherence. Other summarization systems (e.g. Marcu, 1999; Wolf and Gibson, 2005) integrate a more sophisticated model

Title:	former President Carter's international activities
Query:	Describe former President Carter's international efforts including activities of the Carter Center.

Fig. 1 A DUC 2006 topic (D0650E).

of coherence in the content selection process, but they require high level semantic annotation which can (for now) only be achieved manually.

2.2 Task definition

We used the Document Understanding Conference (DUC) 2006 data set for training, and the DUC 2005 data set for testing.[1] This is possible because both data sets are similar. The task at hand is to automatically generate a summary of a maximum of 250 words, given a *topic*. A topic consists of a title, a query, and a set of source documents. The summary should answer the query, using the source documents. An example of a topic is given in Fig. 1. The DUC 2006 document set consists of 50 topics with 25 source documents each. The DUC 2005 document set consists of 50 topics with 25–50 source documents each (approximately 32 on average).

The summarization task is given to professional human summarizers as well as automatic summarization systems. The human summaries are used as *reference summaries* for evaluating *candidate summaries* (i.e., generated summaries). Each DUC 2005 topic has six corresponding reference summaries; each DUC 2006 topic has four. We use Rouge-2 recall (i.e. bigram recall with respect to reference summaries) and Rouge-SU4 recall (skip bigram recall) as performance metrics for evaluation (Lin, 2004), because these metrics were also used (with the same configuration) at DUC 2005 and DUC 2006. Although Rouge metrics provide only a partial evaluation of a summarization system, they are reproducible and repeatable for different system configurations in an objective manner since they require no manual intervention.

To measure if one summarization algorithm performs better (or worse) than another with a particular metric, we count the number of topics for which it outperformed the other, and vice versa. Then, an approximate randomization test is run to measure statistical significance (Noreen, 1989).

2.3 A framework for summarization

Our goal is to investigate new methods for content selection. Nonetheless, the evaluation methods used are designed to measure the quality of abstracts, and require a

[1] Available from `http://duc.nist.gov`

full summarization system. We briefly describe the summarization system, and then focus on the content selection components. The summarization system consists of the following components.

Segmentation. The source documents as well as the query are segmented into sentences. The document name, the paragraph number and sentence number are associated with each sentence as meta-information. The document name can later be used to detect whether sentences are from the same document, or whether they are query sentences. Paragraph boundaries are derived from annotations provided with the source documents. The segmenter also attempts to remove meta data from the text, such as the date and location of publication. These meta data are not part of the running text and may introduce noise in the summary.

Feature extraction. The source text and the query are processed and converted to a feature graph to prepare for content selection. Multiple modules may be used in parallel so that multiple graphs are generated. This may include coherence analysis, measuring redundancy, etc. The generated graphs are integrated into a combined graph, as described later in this chapter.

Salience estimation. A salience value is derived for each sentence from the (possibly combined) feature graph.

Presentation. A summary is created using the most salient content units, up to the word limit of 250 words. If adding the next-salient sentence would cause the word limit to be exceeded, no more sentences are added. Where possible, the linear ordering of the sentences in the source text is retained. If the summary contains sentences from multiple source documents, sentences from the document containing the largest number of sentences are presented first. Although the ordering of the sentences may be important for readability, it has little effect on Rouge scores.

In the next section, we compare different methods for *feature extraction* and *salience estimation*. Across these experiments, the components of *segmentation* and *presentation* remain unchanged.

2.4 Query-based summarization

Below, we describe a number of systems based on our summarization framework, starting with a rudimentary summarization system, and adding features to build increasingly sophisticated systems. The modular summarization framework allows for the flexibility to add feature graphs or replace the salience estimation algorithm.

Wherever parameter optimization is used, the DUC 2006 data set is used for this purpose. The results are then validated with the DUC 2005 data set.

2.4.1 Query-relevance

A simple form of query-based summarization is to determine sentence salience by measuring its cosine similarity with the query. The sentences most similar to the query are presented as a summary. This constitutes a competitive baseline system for query-based summarization. The graph used for salience estimation is the graph in which each candidate sentence is related to each query sentence, and the strength of this relation is the cosine similarity of the two sentences. The sentences closest to a query sentence are then included in the summary. The cosine similarity graph is generated in three steps:

1. the words of all sentences are stemmed using Porter's stemmer (Porter, 2001);
2. the inverse document frequency (IDF) is calculated for each word;
3. the cosine similarity of each candidate sentence and each query sentence is calculated using the $tf \cdot idf$ weighting scheme.

Stemming is a way to normalize morphological variation. For example, the words *cause* and *causes* are different words but refer to the same concept. Stemming allows us to treat both forms in the same way.

The inverse document frequency is used to assign a greater weight to words which occur in fewer sentences. Rare words typically characterize the sentence they appear in to a greater extent than frequent words. IDF is a way to account for this (Spärck Jones, 1972).

IDF calculation requires statistics of the language to determine the frequency of words. For calculating the IDF values of the words in source documents, we use the statistics of the set of all source documents in the topic. It is inappropriate to use the same statistics to weight the query words, because there is a mismatch between the language use in the query and in the source documents. For instance, queries frequently use phrases such as 'Discuss ...' or 'Describe ...'. These words have a low frequency in the source documents, and thus would be assigned a high IDF value, although they are hardly descriptive if they appear in the query. On the other hand, a single query is too short to draw useful statistics from. Therefore, the IDF values for query terms are calculated from the set of sentences from all DUC 2006 queries, while the IDF values for source documents are calculated from the set of source document sentences specific for the topic.

The query-relevance graph (called δ_q) is defined by a function determining the strength of the relation between two sentences:

$$\begin{aligned} \delta_q(i,j) &= cosim(i,j) && \text{, if } i \in Q;\ j \in S \\ \delta_q(i,j) &= 0 && \text{, otherwise} \end{aligned} \tag{1}$$

where $\delta_q(i,j)$ is the strength of the relation between sentences i and j; Q is the set of query sentences; S is the set of candidate sentences; $cosim(i,j)$ is the cosine similarity of sentences i and j. The strength of a relation is a value in the range of 0 (no relation) to 1 (a strong relation).

The query-relevance $R_q(j)$ of a sentence j is then calculated by taking the greatest strength of any relation of j to a query sentence:

$$R_q(j) = \max_{q \in Q}(\delta_q(q,j)) \tag{2}$$

where $R_q(j)$ is the salience of sentence j; Q is the set of query sentences.

A summary is then generated from the most salient sentences.

2.4.2 Contextual relevance

The *cohesion graph* (δ_c) is added as a feature graph for calculating contextual relevance. This graph is constructed the same way the query-relevance graph is constructed, except that it relates candidate sentences of the same document, rather than query sentences and candidate sentences.

The graphs δ_q and δ_c are integrated into a single multi-graph Δ_{q+c}. A multi-graph is a graph that can have two edges between the same two sentences, expressing simultaneous relations. As a result, not a single relation but a set of relations hold between two sentences, and each relation may have a different strength between 0 and 1. The integrated graph is expressed as follows.

$$\Delta_{q+c}(i,j) = \left\{ w_q\delta_q(i,j), w_c\delta_c(i,j) \right\} \tag{3}$$

where $\Delta_{q+c}(i,j)$ is a set of values, each representing the strength of an edge from i to j in the multi-graph Δ_{q+c}. The values of $w_q, w_c \in [0..1]$ are weighting factors. The smaller w_q and the greater w_c, the greater the relative importance of indirect evidence of relevance. A greater w_q (relative to w_c) results in selecting more sentences which can be directly related to the query. A greater w_c represents a greater bias toward sentences which are relevant indirectly, and which may increase the coherence of the summary as a whole.

The salience estimation algorithm calculates the salience of each sentence, given a graph of relations between sentences. A relation from sentence X to sentence Y increases the relevance (and therefore the salience) of Y if X is relevant. This immediately poses a problem if X is a candidate sentence, because the relevance of Y depends on the relevance of X, which itself is not yet calculated. Literature provides several solutions (Mani and Bloedorn, 1997; Erkan and Radev, 2004), which have in common that they iteratively recalculate the salience of a sentence in a graph from the salience of neighboring sentences. Following this process, salience is calculated as follows.

1. Initiate the salience of all candidate sentences (source document sentences) at 0. The salience of query sentences is initiated at 1.
2. Recalculate the salience of each candidate sentence, using the feature graphs and the salience of neighboring (i.e. related) sentences. Salient sentences increase the salience of their neighbors.

3. Repeat step 2 untill the sum of changes in salience in the last iteration falls below a certain (pre-defined) threshold.

We used two salience estimation algorithms, *normalized centrality* and *probabilistic relevance*. They differ in how they recalculate relevance (step 2).

The first, based on Erkan and Radev (2004), recalculates the salience by dividing the salience of each sentence among its neighboring sentences. Because the sum of salience values of all sentences remains approximately constant (amounts of salience are just "passed on" from one sentence to the next), we call this *normalized centrality*.

The *probabilistic relevance* algorithm regards the feature graph as a probabilistic semantic network. The salience of a sentence represents the probability that the sentence is relevant, and a relation from sentence X to Y is the probability that Y is relevant, given X is relevant.

Normalized centrality. The result of the normalized centrality process is a *centrality* value for each sentence, which represents the salience of the sentence. The basic idea behind this algorithm is that at each iteration, the centrality value of each sentence is distributed among its related sentences. For instance, if a sentence has three outgoing relations, the centrality value is divided among these three sentences. Conversely, the new centrality value of a sentence at the next iteration is calculated from the sum of "centrality" received from sentences to which it has an incoming relation.

We use the symbol μ_j to indicate the centrality of sentence j. The value of $\mu_j(t)$ is the centrality of sentence j at iteration $t \geq 0$, which is calculated as follows:

$$\begin{aligned}
\mu_j(t) &= 1 && \text{, if } j \in Q \\
\mu_j(0) &= 0 && \text{, if } j \in S \\
\mu_j(t+1) &= \frac{d}{\|D\|} + (1-d)\sum_{i \in D} x(i,j) && \text{, if } j \in S \\
x(i,j) &= \sum_{r \in \Delta_{q+c}(i,j)} r \cdot \mu_i(t) \cdot degree(i)^{-1}
\end{aligned} \tag{4}$$

where $D = Q \cup S$; and $\Delta_{q+c}(i,j)$ is the set of edges between i and j in the relevance graph.

By definition, query sentences have a centrality of 1. In the first iteration, the centrality of all other sentences is initialized to 0. At each iteration, the normalized centrality of each sentence is distributed to its neighbor sentences (the sentence's related sentences). The constant d is a small value which is required in order to guarantee that the algorithm converges under all circumstances by giving each sentence a small a priori non-zero centrality.[2] The degree of a sentence i in the graph is measured as the sum of the weights of the outgoing edges:

[2] Throughout this section, the value of 0.15 is used, as suggested in Erkan and Radev (2004), but the actual value of d has no effect on the final centrality ranking as long as it is non-zero.

$$degree(i) = \sum_{k \in D} \sum_{(r \in \Delta_{q+c}(i,k))} r \tag{5}$$

A characteristic of this algorithm is that across iterations, the sum of the centrality of all sentences stays approximately the same: centrality is treated as a kind of commodity which may be transferred from one sentence to the next, but no centrality is created or lost. The only exceptions to this rule are the query sentences (which always have a centrality of one, and therefore may "create" centrality) and the constant d, which causes a small centrality value to be assigned to each sentence.

The result is a centrality (i.e., salience) value μ between 0 and 1 associated with each passage. The content units with the highest salience values are selected for inclusion in the summary. In this configuration, normalization cancels out the effect of graph weighting: changing the graph weights w_q and w_c (cf. equation 3) does not affect the summaries in any way because the relevance distribution is normalized and the sets of sentences with outgoing edges in δ_q and δ_c are disjunct.

Probabilistic relevance. In the probabilistic approach, relations between sentences are interpreted as probabilities. The strength of a relation is the probability that the target sentence is relevant, provided that the origin sentence of the relation is relevant. In this algorithm, the salience calculation of a sentence at each iteration depends only on the salience values of its related sentences and the strengths of these relations. This is unlike normalized centrality, where a sentence has to "compete" for a related sentence's centrality, since centrality is distributed among related sentences proportional to the relation strengths.

The query sentences are relevant by definition (although they are not actually included in the summary):

$$\nu_j(t) = 1 \qquad , \text{if } j \in Q$$

If a candidate sentence has only one incoming relation, its probabilistic relevance in the next iteration is the strength of the relation multiplied by the relevance of the origin sentence of the relation:

$$\begin{aligned} \nu_j(0) &= 0 && , \text{if } j \in S \\ \nu_j(t+1) &= r \cdot \nu_i(t) \cdot y && , \text{if } j \in S; t \geq 0 \end{aligned} \tag{6}$$

where $\nu_j(t)$ is the probabilistic relevance value of sentence j at iteration t, and r is the strength of the relation. The value of y is the *decay value*, a global constant in the range $\langle 0..1 \rangle$. The constant y has a function similar to the constant d in normalized centrality: it is necessary to ensure that the relevance value converges.

If a sentence has more incoming relations, its relevance value depends on the relevance of the origin sentences of the relations. If $P(j \mid i)$ denotes the (conditional) probability that j is relevant, provided that i is relevant, then $P(\neg j \mid i) = 1 - P(j \mid i)$ is the probability that i is *not* relevant, provided that i is relevant. If we read all relation strengths as the conditional probability that the target sentence is *not* relevant (the *inverse probability*), we can combine multiple probabilities by multiplying their inverse probabilities, and then again taking the inverse of the result to get the combined probability:

$$P(i) = 1 - \prod_{j}(1 - P(i \mid j)) \tag{7}$$

Combining eq. 6 and 7, the relevance of a sentence j is calculated at each iteration as:

$$\nu_j(t+1) = 1 - \prod_{i \in Q \cup S} \prod_{r \in \Delta_{q+c}(i,j)} (1 - r \cdot \nu_i(t) \cdot y)$$

The relation set $\Delta_{q+c}(i,j)$ is the result of the union of the graphs δ_q and δ_c, and their weights w_q and w_c. The optimal weight values are estimated by measuring Rouge-2 performance for different weight values. First, w_q is incremented in steps of 0.1 from 0 to 1 with $w_c = 1$, and then w_c is incremented in steps of 0.1 from 0 to 1 with $w_q = 1$. The optimal weight settings are $w_q = 1$; $w_c = 0.1$.

2.4.3 Redundancy-aware summarization

One of the assumptions often made implicitly in the design of single-document summarization systems, is that the source document does not contain redundancy. Consequently, there is no risk of including a sentence in the summary which does not contain any new information. This changes when a summary is generated from multiple source documents, where non-redundancy of sentences from different documents cannot be taken for granted. The content selection procedures outlined previously concentrate entirely on relevance, not on redundancy. However, in multi-document summarization, presented content should be relevant to the query and novel with respect to what is already mentioned in the summary. In other words, salience comprises both relevance and novelty.

To accommodate representing novelty, we extended the model with a redundancy feature graph Υ which is used in addition to the previously mentioned relevance feature graph Δ. Similarly to relevance, redundancy relations have a strength in the range $[0..1]$. The strength of a redundancy relation between two sentences expresses the likelihood that a sentence is redundant, given the fact that another sentence is relevant.

Although redundancy comes into play only during sentence selection, this way of modeling redundancy makes it possible to generate the redundancy graph in parallel with the relevance graph generation. That is, *before* the actual sentence selection, or even before the query is formulated. The actual novelty calculation still has to take place during sentence selection, but this is just a simple calculation while the redundancy graph may be the result of sophisticated processing algorithms. The redundancy graph generation algorithms do not need specific knowledge of summarization and can focus on their isolated task, which is to calculate the probability that a sentence i is redundant, provided that a sentence j is relevant, for each pair of sentences i, j.

The redundancy of sentence j, given sentence i, is defined by $\delta_r(i,j)$. The form of the redundancy graph is identical to that of the relevance graph. The strengths of relations in the redundancy feature graph δ_r are defined as follows:

$$\begin{array}{ll} \delta_r(i,j) = cosim(i,j) & \text{, if } i,j \in S;\ doc(i) \neq doc(j) \\ \delta_r(i,j) = 0 & \text{, otherwise} \end{array} \tag{8}$$

The redundancy-aware summarization system uses a set of redundancy feature graphs Υ for determining salience of sentences, in addition to the relevance feature graphs Δ:

$$\begin{aligned} \Delta_{q+c+r}(i,j) &= \left\{w_q \cdot \delta_q(i,j), w_c \cdot \delta_c(i,j), w_{r\Delta} \cdot \delta_r(i,j)\right\} \\ \Upsilon_r(i,j) &= \{w_{r\Upsilon} \cdot \delta_r(i,j)\} \end{aligned} \tag{9}$$

where δ_q, δ_c and δ_r are the query-relevance graph, the cohesion graph, and the redundancy graph respectively. The set of relations between sentences i and j is represented by $\Delta_{q,c,r}(i,j)$ (relevance) and $\Upsilon_r(i,j)$ (redundancy). Since redundancy implies 'relatedness', we regard a redundancy graph as a special case of a relevance graph. Therefore, δ_r is not only included in Υ_r but also in Δ_{q+c+r} (with weights $w_{r\Upsilon}$ and $w_{r\Delta}$ respectively).

The calculation of redundancy-adjusted salience was inspired by Carbonell and Goldstein (1998). First, the relevance of each sentence is calculated using Δ_{q+c+r}. Then, the novelty is calculated – novelty is the reciprocal of redundancy. If two sentences are redundant, this affects only the novelty of the least-relevant of the two. The implication of this is that a redundancy relation may cause a sentence to be downranked, but only if it is redundant with respect to a higher ranking sentence (which was already preferred over the downranked sentence during sentence selection). The stronger the redundancy relation, the greater the reduction of novelty. Novelty is calculated as follows:

$$\begin{aligned} N(j) &= \prod_{i \in F_j} \prod_{r \in \Upsilon_r(i,j)} (1 - r \cdot R(i)) \\ F_j &= \{k : S \mid R(k) > R(j)\} \end{aligned} \tag{10}$$

where $N(j)$ is a value in the range $[0..1]$, representing the novelty of sentence j; $\Upsilon_r(i,j)$ is a set of redundancy relations, expressing the redundancy of j given i; F_j is the set of content units more relevant than j. The function $R(i)$ denotes the relevance of sentence i, as previously calculated.

Now, the redundancy-adjusted salience can be calculated as the product of relevance and novelty:

$$\sigma_j = R(j) \cdot N(j) \tag{11}$$

Table 1 Performance on DUC 2006 data: Rouge scores, and the system rank among 36 systems (between brackets) if it had participated in DUC 2006.

System	Rouge-2	Rouge-SU4
Query-relevance	.08180 (11)	.1384 (11)
Normalized centrality	.08195 (11)	.1362 (11)
Probabilistic relevance	.08884 (3)	.1432 (7)
Redundancy-aware normalized centrality	.09294 (2)	.1496 (2)
Redundancy-aware probabilistic redundancy	.09305 (2)	.1501 (2)
Best DUC 2006 submission	.09505 (-)	.1546 (-)

Table 2 Percentage of DUC 2006 topics (Rouge-2/Rouge-SU4) for which one system (rows) beat another (columns). Note that percentages do not add up to 100 if both systems receive the same score for at least one topic.

%	(a)	(b)	(c)	(d)	(e)
(a) Query-relevance	–	50/52	34[a]/28[a]	30[a]/28[a]	26[a]/26[a]
(b) Normalized centrality	46/48	–	34[a]/36[b]	38[b]/34[a]	30[a]/24[a]
(c) Probabilistic relevance	64[a]/70[a]	66[a]/62[b]	–	56/58	44/50
(d) Redundancy-aware normalized centrality	66[a]/66[a]	60[b]/62[a]	42/42	–	30[a]/30[a]
(e) Redundancy-aware probabilistic relevance	70[a]/72[a]	68[a]/72[a]	48/46	64[a]/68[a]	–

[a] Significant at $p < 0.01$.
[b] Significant at $p < 0.05$.
[c] Significant at $p < 0.1$.

where σ_j is the redundancy-adjusted salience of sentence j. The calculation of σ_j ensures that:

- if one content unit is selected, all content units redundant to that unit are less likely to be selected: if two content units are redundant with respect to each other, the salience of the least-relevant content unit is reduced;
- redundancy of a content unit does not prevent relevance to propagate: a redundant content unit may still be relevant.

We determined the graph weights by starting from the optimal values for w_q and w_c, as specified in section 2.4.2. The remaining weights are determined by means of a similar procedure as in section 2.4.2: first, $w_{r\Delta}$ is incremented in steps of 0.1 from 0 to 1 with $w_{r\Upsilon} = 0$, and then $w_{r\Upsilon}$ is incremented in steps of 0.1 from 0 to 1 without changing the other weights.

For the normalized centrality algorithm, the resulting optimal weight settings are $w_q = 1$; $w_c = 1$ and $w_{r\Upsilon} = 0$; $w_{r\Delta} = 1$. Increasing the value of $w_{r\Upsilon}$ has no effect on the quality of the summaries. For the probabilistic relevance algorithm, the resulting optimal weight settings are $w_q = 1$; $w_c = 0.1$; $w_{r\Delta} = 0.2$; $w_{r\Upsilon} = 1$.

Table 3 Performance on DUC 2005 data: Rouge scores, and the system rank among 32 systems (between brackets) if it had participated in DUC 2005.

System	Rouge-2	Rouge-SU4
Query-relevance	.07056 (3)	.1260 (5)
Normalized centrality	.07210 (2)	.1262 (5)
Probabilistic relevance	.06923 (5)	.1247 (6)
Redundancy-aware normalized centrality	.07230 (2)	.1291 (3)
Redundancy-aware probabilistic relevance	.07362 (1)	.1300 (2)
Best DUC 2005 submission	.07251 (-)	.1316 (-)

Table 4 Percentage of DUC 2005 topics (Rouge-2/Rouge-SU4) for which one system (rows) beat another (columns). Note that percentages do not add up to 100 if both systems receive the same score for at least one topic.

%	(a)	(b)	(c)	(d)	(e)
(a) Query-relevance	–	46/44	42/42	50/50	40[c]/40[c]
(b) Normalized centrality	52/54	–	50/34[a]	50/54	38[b]/34[a]
(c) Probabilistic relevance	54/58	50/66[a]	–	58[b]/64[a]	36[c]/42
(d) Redundancy-aware normalized centrality	44/44	46/44	38[b]/36[a]	–	30[a]/30[a]
(e) Redundancy-aware probabilistic relevance	58[c]/60[c]	60[b]/66[a]	54[c]/54	60[a]/70[a]	–

[a] Significant at $p < 0.01$.
[b] Significant at $p < 0.05$.
[c] Significant at $p < 0.1$.

2.5 Results

Table 1 shows the Rouge scores for each of the summarization systems. Table 2 gives an overview of the differences in performance between the systems. The query-relevance system was beaten by every other system ($p < 0.01$) except the normalized centrality system. The (redundancy-aware) probabilistic relevance system outperformed the (redundancy-aware) normalized centrality system ($p < 0.05$). The introduction of redundancy significantly improved results in the normalized centrality system ($p < 0.05$), but not in the probabilistic relevance system. The redundancy-aware probabilistic relevance system is the only system which beats all systems except the (non redundancy-aware) probabilistic relevance system ($p < 0.01$).

2.6 Validating the results

The previous section outlined a comparison of different configurations of the summarization framework. However, the way the graph weight configurations are determined implies that the weights are tailored to the DUC 2006 data set. As a result, there is a risk that the weights are overfitted to this particular set. In order to validate the results, we ran the experiments on the DUC 2005 data set with the graph weight configurations determined in section 2.4.

Table 3 shows the average Rouge-2 and Rouge-SU4 scores achieved with the DUC 2005 corpus. Table 4 shows an overview of the pair-wise significance tests. The redundancy-aware probabilistic relevance system significantly outperforms all other systems when Rouge-2 is used ($p < 0.1$), and all except the redundancy-aware normalized centrality system according to Rouge-SU4. This system would have ranked first (Rouge-2) or second (Rouge-SU4) if it had participated in DUC 2005. The overall picture confirms the preliminary results in section 2.5 with respect to the differences between normalized centrality and the probabilistic relevance: the latter system outperformed the first (Rouge-2: $p < 0.01$; although no significant differences were measured using Rouge-SU4), and the redundancy-aware variant of the probabilistic relevance system also outperformed the redundancy-aware normalized centrality system ($p < 0.01$, Rouge-2 and Rouge-SU4). However, introducing redundancy did not generally improve the results. Only in the probabilistic relevance system, we found a significant Rouge-2 improvement ($p < 0.1$) as a result of the added redundancy graphs.

An interesting observation is that the probabilistic relevance system has the lowest average scores, but still beats the normalized centrality system in terms of the number of summaries for which the Rouge-SU4 score was greater. Apparently, the probabilistic relevance system usually (in 66% of the cases) beat the normalized centrality system, but there were a few cases in which the score of the probabilistic system was considerably lower. This raises the question which system is better: the one which receives the greatest average score, or the one which most frequently produces the better summary?

Note also that it is not guaranteed that the combination of graph weights that leads to the best performance has been found. Apart from the risk of overfitting, the number of possible graph weight combinations is infinite and a greater number of graphs makes it more difficult to find the best combination of weights. A future extension would use machine learning methods such as genetic algorithms that are better suited to find the optimal solution. Despite this, we already achieved good result with the current system, using limited optimization.

3 Sentence fusion

Question-answering systems conventionally follow the strategy of retrieving possible answers from a text collection, ranking these answers according to some criteria of goodness, and outputting the top-ranked answer. Many current QA systems even rely on various parallel answer-finding strategies, each of which may produce an n-best list of answers (e.g. Maybury, 2004). However, the underlying assumption that a single complete answer can be pinpointed in the text collection is questionable at least. Especially if the question is of the *open* type rather than the *factoid* type (where the answer typically is some named entity), relevant parts of the answer may be scattered among the candidate answers on the n-best list. For example, in response to the open question *What causes RSI?* one candidate answer may be:

RSI can be caused by repeating the same sequence of movements many times an hour or day.

However, another candidate could be:

RSI is generally caused by a mixture of poor ergonomics, stress and poor posture.

Clearly neither of these two examples constitutes a single complete answer on its own. A more satisfying alternative would require a fusion of the two partial answers into a more complete one such as:

RSI can be caused by a mixture of poor ergonomics, stress, poor posture and by repeating the same sequence of movements many times an hour or day.

This notion of *sentence fusion* was first introduced by Barzilay (2003). Sentence fusion is a text-to-text generation application, which given two related sentences, outputs a single sentence expressing the information shared by the two input sentences. The process of sentence fusion comprises four major steps:

1. **Linguistic analysis** The input sentences are tokenized and syntactically parsed.
2. **Alignment** The syntax trees are aligned by matching syntactic nodes with similar meaning.
3. **Merging** The syntax trees are combined into a single tree.
4. **Generation** The surface string for the output sentence is generated.

This was originally applied in the context of multi-document summarization, where it was used to combine similar sentences extracted from different documents in order to increase compression and avoid redundancy. However, as we proposed earlier (Marsi and Krahmer, 2005a,b), sentence fusion may be applied in a question-answering context as well for the purpose of combining candidate answers.

Ideally we use an off-the-shelf sentence fusion system, plug it into an existing QA system, and run some evaluations in order to measure its contribution to overall answer quality. In reality, there are no readily available sentence fusion systems yet and there are at least a number of open research questions. The work reported here reviews our earlier work on sentence fusion. It concerns the implementation and evaluation of models for alignment, merging and generation in Dutch. We start in section 3.1 with the basic question whether it is possible at all to reliably align sentences. After defining the alignment task, we describe the construction of a parallel monolingual corpus consisting of manually aligned syntax trees, discussing results on inter-annotator agreement. Section 3.2 then addresses automatic alignment. We present our algorithm for automatic tree alignment as well as its evaluation on the parallel corpus. Next, section 3.3 describes exploratory work on simple methods for the merging and generation steps in the sentence fusion process, including an evaluation test. We end with a discussion and description of more recent follow-up research in section 3.4.

3.1 Data collection and Annotation

3.1.1 General approach to alignment

Alignment has become standard practice in data-driven approaches to machine translation (e.g. Och and Ney, 2000). Initially work focused on word-based alignment, but more recent research also addresses alignment at the higher levels (substrings, syntactic phrases or trees), e.g., Gildea (2003). The latter approach seems most suitable for current purposes, where we want to express that a sequence of words in one sentence is related to a non-identical sequence of words in another sentence (a paraphrase, for instance). However, if we allow alignment of arbitrary substrings of two sentences, then the number of possible alignments grows exponentially to the number of tokens in the sentences, and the process of alignment – either manually or automatically – may become infeasible. An alternative, which seems to occupy the middle ground between word alignment on the one hand and alignment of arbitrary substrings on the other, is to align syntactic analyses. Here, following Barzilay (2003), we will align sentences at the level of **dependency structures**.

Rather than a binary choice (align or not), one might want to distinguish more fine-grained relations such as overlap (if two phrases share some but not all of their content), paraphrases (if two phrases express the same information in different ways), entailments (if one phrase entails the other, but not vice versa), etc. Unlike Barzilay (2003), we therefore not only align similar nodes, but also label the node alignments according to a small set of semantic similarity relations, which will be defined in the next section. This additional information allows for alternative ways of fusing sentences, as described in section 3.3, which are especially interesting in the context of QA.

3.1.2 Task definition

A dependency analysis of a sentence S yields a labeled directed graph $D = \langle V, E \rangle$, where V (vertices) are the nodes, and E (edges) are the dependency relations. For each node v in the dependency structure for a sentence S, we define $\textsc{str}(v)$ as the substring of all tokens under v (i.e., the composition of the tokens of all nodes reachable from v). For example, the string associated with node *persoon* in Fig. 2 is *heel veel serieuze personen* ('very many serious persons').

An alignment between sentences S and S' pairs nodes from the dependency graphs for both sentences. Aligning node v from the dependency graph D of sentence S with node v' from the graph D' of S' indicates that there is a relation between $\textsc{str}(v)$ and $\textsc{str}(v')$, i.e., between the respective substrings associated with v and v'. We distinguish five potential, mutually exclusive, relations between nodes (with illustrative examples):

1. v **equals** v' iff $\textsc{str}(v)$ and $\textsc{str}(v')$ are literally identical (abstracting from case and word order)

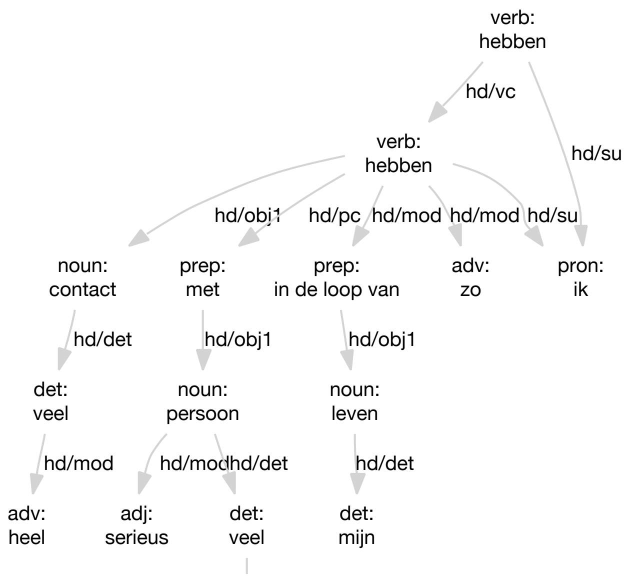

Fig. 2 Example dependency structure for the sentence *Zo heb ik in de loop van mijn leven heel veel contacten gehad met heel veel serieuze personen.* (lit. 'Thus have I in the course of my life very many contacts had with very many serious persons').

Example: "a small and a large boa-constrictor" equals "a large and a small boa-constrictor";

2. v **restates** v' iff $\text{STR}(v)$ is a paraphrase of $\text{STR}(v')$ (same information content but different wording),
   Example: "a drawing of a boa-constrictor snake" restates "a drawing of a boa-constrictor";
3. v **specifies** v' iff $\text{STR}(v)$ is more specific than $\text{STR}(v')$,
   Example: "the planet B 612" specifies "the planet";
4. v **generalizes** v' iff $\text{STR}(v')$ is more specific than $\text{STR}(v)$,
   Example: "the planet" generalizes "the planet B 612";
5. v **intersects** v' iff $\text{STR}(v)$ and $\text{STR}(v')$ share some informational content, but also each express some piece of information not expressed in the other,
   Example: "Jupiter and Mars" intersects "Mars and Venus"

Note that there is an intuitive relation with entailment here: both *equals* and *restates* can be understood as mutual entailment (i.e., if the root nodes of the analyses corresponding S and S' stand in an equal or restate relation, S entails S' and S' entails S), if S *specifies* S' then S also entails S' and if S *generalizes* S' then S is entailed by S'.

An alignment between S and S' can now formally be defined on the basis of the respective dependency graphs $D = \langle V, E\rangle$ and $D' = \langle V', E'\rangle$ as a graph $A = \langle V_A, E_A\rangle$, such that

$$E_A = \{\langle v, l, v' \rangle \mid v \in V \ \& \ v' \in V' \ \& \ l(\textsc{str}(v), \textsc{str}(v'))\},$$

where l is one of the five relations defined above. The nodes of A are those nodes from D en D' which are aligned, formally defined as

$$V_A = \{v \mid \exists v' \exists l \langle v, l, v' \rangle \in E_A\} \cup \{v' \mid \exists v \exists l \langle v, l, v' \rangle \in E_A\}$$

3.1.3 Corpus

For evaluation and parameter estimation we have developed a **parallel monolingual corpus** consisting of two different Dutch translations of the French book "Le petit prince" (*the little prince*) by Antoine de Saint-Exupéry (published 1943), one by Laetitia de Beaufort-van Hamel (1966) and one by Ernst van Altena (2000). The texts were automatically tokenized and split into sentences, after which errors were manually corrected. Corresponding sentences from both translations were manually aligned; in most cases this was a one-to-one mapping but occasionally a single sentence in one version mapped onto two sentences in the other. Next, the **Alpino** parser for Dutch (e.g. Bouma et al., 2001) was used for part-of-speech tagging and lemmatizing all words, and for assigning a dependency analysis to all sentences. The POS labels indicate the major word class (e.g. *verb*, *noun*, *pron*, and *adv*). The dependency relations hold between tokens and are the same as used in the Spoken Dutch Corpus (see e.g. van der Wouden et al., 2002). These include dependencies such as *head/subject*, *head/modifier* and *coordination/conjunction*. See Fig. 2 for an example. If a full parse could not be obtained, Alpino produced partial analyses collected under a single root node. Errors in lemmatization, POS tagging, and syntactic dependency parsing were not subject to manual correction.

3.1.4 Results

All text material was aligned by two annotators. They started doing the first ten sentences of Chapter One together in order to get a feel for the task. They continued with the remaining sentences from Chapter One individually. The total number of nodes in the two translations of the chapter was 445 and 399 respectively. Inter-annotator agreement was calculated for two aspects: alignment and relation labeling. With respect to alignment, we calculated the precision, recall and F-score (with $\beta = 1$) on aligned node pairs as follows:

$$precision(A_{real}, A_{pred}) = \frac{|A_{real} \cap A_{pred}|}{|A_{pred}|}$$

$$recall(A_{real}, A_{pred}) = \frac{|A_{real} \cap A_{pred}|}{|A_{real}|}$$

$$F\text{-}score = \frac{2 \times precision \times recall}{precision + recall}$$

where A_{real} is the set of all real alignments (the reference or golden standard), A_{pred} is the set of all predicted alignments, and $A_{pred} \cap A_{real}$ is the set of all correctly predicted alignments. For the purpose of calculating inter-annotator agreement, one of the annotations (A_1) was considered the 'real' alignment, the other (A_2) the 'predicted'. The results are summarized in Table 5 in column (A_1,A_2).

Table 5 Interannotator agreement with respect to alignment between annotators 1 and 2 before (A_1,A_2) and after $(A_{1'},A_{2'})$ revision, and between the consensus and annotator 1 $(A_c,A_{1'})$ and annotator 2 $(A_c,A_{2'})$ respectively.

	(A_1,A_2)	$(A_{1'},A_{2'})$	$(A_c,A_{1'})$	$(A_c,A_{2'})$
#real:	322	323	322	322
#pred:	312	321	323	321
#correct:	293	315	317	318
precision:	.94	.98	.98	.99
recall:	.91	.98	.98	.99
F-score:	.92	.98	.98	.99

Next, both annotators discussed the differences in alignment, and corrected mistaken or forgotten alignments. This improved their agreement as shown in column $(A_{1'},A_{2'})$. In addition, they agreed on a single consensus annotation (A_c). The last two columns of Table 5 show the results of evaluating each of the revised annotations against this consensus annotation. The F-score of .96 can therefore be regarded as the upper bound on the alignment task.

Table 6 Inter-annotator agreement with respect to alignment **relation labeling** between annotators 1 and 2 before (A_1,A_2) and after $(A_{1'},A_{2'})$ revision , and between the consensus and annotator 1 $(A_c,A_{1'})$ and annotator 2 $(A_c,A_{2'})$ respectively.

	(A_1,A_2)	$(A_{1'},A_{2'})$	$(A_c,A_{1'})$	$(A_c,A_{2'})$
precision:	.86	.96	.98	.97
recall:	.86	.95	.97	.97
F-score:	.85	.95	.97	.97
κ:	.77	.92	.96	.96

In a similar way, the agreement was calculated for the task of labeling the alignment relations. Results are shown in Table 6, where the measures are *weighted* precision, recall and F-score. For instance, the precision is the weighted sum of the separate precision scores for each of the five relations. The table also shows the κ-score, which is another commonly used measure for inter-annotator agreement (Carletta, 1996). Again, the F-score of .97 can be regarded as the upper bound on the relation labeling task.

We think these numbers indicate that the labeled alignment task is well defined and can be accomplished with a high level of inter-annotator agreement.

3.2 Automatic alignment

3.2.1 Tree alignment algorithm

The tree alignment algorithm is based on Meyers et al. (1996), and similar to that used in Barzilay (2003). It calculates the match between each node in **dependency tree** D against each node in dependency tree D'. The score for each pair of nodes only depends on the similarity of the words associated with the nodes and, recursively, on the scores of the best matching pairs of their descendants. For an efficient implementation, dynamic programming is used to build up a score matrix, which guarantees that each score will be calculated only once.

Given two dependency trees D and D', the algorithm builds up a score function $S(v,v')$ for matching each node v in D against each node v' in D', which is stored in a matrix M. The value $S(v,v')$ is the score for the best match between the two subtrees rooted at v in D and at v' in D'. When a value for $S(v,v')$ is required, and is not yet in the matrix, it is recursively computed by the following formula:

$$S(v,v') = max \begin{cases} \textsc{TreeMatch}(v,v') \\ max_{i=1,\ldots,n}\, S(v_i,v') \\ max_{j=1,\ldots,m}\, S(v,v'_j) \end{cases}$$

where $v_1,\ldots,v_n$ denote the children of v and $v'_1,\ldots,v'_m$ denote the children of v'. The three terms correspond to the three ways that nodes can be aligned: (1) v can be directly aligned to v'; (2) any of the children of v can be aligned to v'; (3) v can be aligned to any of the children of v'. Notice that the last two options imply skipping one or more edges, and leaving one or more nodes unaligned.[3]

The function $\textsc{TreeMatch}(v,v')$ is a measure of how well the subtrees rooted at v and v' match:

$$\textsc{TreeMatch}(v,v') = \textsc{NodeMatch}(v,v') + \\ \max_{p\,\in\,\mathcal{P}(v,v')} \left[\sum_{(i,j)\,\in\,p} \left(\textsc{RelMatch}(\overrightarrow{v}_i, \overrightarrow{v}'_j) + S(v_i,v'_j) \right) \right]$$

Here $\overrightarrow{v}_i$ denotes the dependency relation from v to v_i. $\mathcal{P}(v,v')$ is the set of all possible pairings of the n children of v against the m children of v', which is the power set of $\{1,\ldots,n\} \times \{1,\ldots,m\}$. The summation ranges over all pairs, denoted by (i,j), which appear in a given pairing $p \in \mathcal{P}(v,v')$. Maximizing this summation thus amounts to finding the optimal alignment of children of v to children of v'.

$\textsc{NodeMatch}(v,v') \geq 0$ is a measure of how well the label of node v matches the label of v'.

[3] In the original formulation of the algorithm by Meyers et al. (1996), there is a penalty for skipping edges.

$\textsc{RelMatch}(\overrightarrow{v}_i, \overrightarrow{v}'_j) \geq 0$ is a measure for how well the dependency relation between node v and its child v_i matches that of the dependency relation between node v' and its child v_j.

Since the dependency graphs delivered by the Alpino parser were usually not trees, they required some modification in order to be suitable input for the tree alignment algorithm. We first determined a root node, which is defined as a node from which all other nodes in the graph can be reached. In the rare case of multiple root nodes, an arbitrary one was chosen. Starting from this root node, any cyclic edges were temporarily removed during a depth-first traversal of the graph. The resulting directed acyclic graphs may still have some amount of structure sharing, but this poses no problem for the algorithm.

3.2.2 Evaluation of automatic alignment

We evaluated the automatic alignment of nodes, abstracting from relation labels, as we have no algorithm for automatic labeling of these relations yet. The baseline is achieved by aligning those nodes which stand in an *equals* relation to each other, i.e., a node v in D is aligned to a node v' in D' iff $\textsc{str}(v) = \textsc{str}(v')$. This alignment can be constructed relatively easily.

The alignment algorithm is tested with the following NODEMATCH function:

$$\textsc{NodeMatch}(v, v') = \begin{cases} 10 \text{ if } \textsc{str}(v) = \textsc{str}(v') \\ 5 \text{ if } \textsc{label}(v) = \textsc{label}(v') \\ 2 \text{ if } \textsc{label}(v) \text{ is a synonym} \\ \quad \text{hyperonym or hyponym} \\ \quad \text{of } \textsc{label}(v') \\ 0 \text{ otherwise} \end{cases}$$

It reserves the highest value for a literal string match, a somewhat lower value for matching lemmas, and an even lower value in case of a synonym, hyperonym or hyponym relation. The latter relations are retrieved from the Dutch part of EuroWordnet (Vossen, 1998). For the RELMATCH function, we simply used a value of 1 for identical dependency relations, and 0 otherwise. These values were found to be adequate in a number of test runs on two other, manually aligned chapters (these chapters were not used for the actual evaluation). In the future we intend to experiment with automatic optimizations.

We measured the alignment accuracy defined as the percentage of correctly aligned node pairs, where the consensus alignment of Chapter One served as the golden standard. The results are summarized in Table 7. In order to test the contribution of synonym and hyperonym information for node matching, performance is measured with and without the use of EuroWordnet. The results show that the algorithm improves substantially on the baseline. The baseline already achieves a relatively high score (an F-score of .56), which may be attributed to the nature of our material: the translated sentence pairs are relatively close to each other and may show a sizeable amount of literal string overlap. The alignment algorithm (without use of EuroWordnet) loses a few points on precision, but improves a lot on recall (a 200% increase with respect to the baseline), which in turn leads to a substantial improvement on the overall F-score. The use of EuroWordnet leads to a small increase (two points) on both precision and recall (and thus to small increase on F-score). Yet, in comparison with the gold standard human score for this task (.95), there is clearly room for further improvement.

Table 7 Precision, recall and F-score on automatic alignment

Alignment :	*Prec* :	*Rec* :	*F-score:*
baseline	.87	.41	.56
algorithm without wordnet	.84	.82	.83
algorithm with wordnet	.86	.84	.85

3.3 Merging and generation

The remaining two steps in the sentence fusion process are merging and generation. In general, **merging** amounts to deciding which information from either sentence should be preserved, whereas **generation** involves producing a grammatically correct surface representation. In order to get an idea about the baseline performance, we explored a simple, somewhat naive string-based approach. Below, the pseudo code is shown for merging two dependency trees in order to get restatements. Given a labeled alignment A between dependency graphs D and D', if there is a **restates** relation between node v from D and node v' from D', we add the string realization of v' as an alternative to those of v.

```
RESTATE($A$)
1 for each edge $\langle v, l, v' \rangle \in E_A$
2       do if $l =$ restates
3             then $\textsc{str}(v) \leftarrow \textsc{str}(v) \vee \textsc{str}(v')$
```

The same procedure is followed in order to get specifications:

```
SPECIFY($A$)
1 for each edge $\langle v, l, v' \rangle \in E_A$
2       do if $l =$ generalizes
3             then $\textsc{str}(v) \leftarrow \textsc{str}(v) \vee \textsc{str}(v')$
```

The generalization procedure adds the option to omit the realization of a modifier that is *not* aligned:

```
GENERALIZE($D$,$A$)
1 for each edge $\langle v,l,v'\rangle \in E_A$
2     do if $l =$ specifies
3         then $\mathrm{STR}(v) \leftarrow \mathrm{STR}(v) \vee \mathrm{STR}(v')$
4 for each edge $\langle v,l,v'\rangle \in E_D$
5     do if $l \in$ MOD-DEP-RELS and $v \notin E_A$
6         then $\mathrm{STR}(v) \leftarrow \mathrm{STR}(v) \vee$ NIL
```

where MOD-DEP-REL is the set of dependency relations between a node and a modifier (e.g. *head/mod* and *head/predm*).

Each procedure is repeated twice, once adding substrings from D into D' and once the other way around. Next, we traverse the dependency trees and generate all string realizations, extending the list of variants for each node that has multiple realizations. Finally, we filter out multiple copies of the same string, as well as strings that are identical to the input sentences.

As expected, many of the resulting variants are ungrammatical, because constraints on word order, agreement or subcategorisation are violated. Following work on statistical surface generation (Langkilde and Knight, 1998) and other work on sentence fusion (Barzilay, 2003), we try to filter out ungrammatical variants with an n-gram language model. The Cambridge-CMU Statistical Modeling Toolkit v2 was used to train a 3-gram model on over 250M words from the Twente Newscorpus, using back-off and Good-Turing smoothing. Variants were ranked in order of increasing entropy.

To gain some insight into the general performance of the merging and generation strategy, we performed a small evaluation test in which two judges independently judged all generated variants in terms of three categories:

1. **Perfect**: no problems in either semantics or syntax;
2. **Acceptable**: understandable, but with some minor flaws in semantics or grammar;
3. **Nonsense**: serious problems in semantics or grammar

Table 8 shows the number of sentences in each of the three categories per judge, broken down in restatements, generalization and specifications. The κ-score on this classification task is .75, indicating a moderate to good agreement between the judges. Roughly half of the generated sentences are perfect, although specifications are somewhat less well-formed.

We think we can conclude from this evaluation that sentence fusion is a viable and interesting approach for producing restatements, generalization and specifications. However, there is certainly further work to do; the procedure for merging dependency graphs should be extended, and the realization model clearly requires more linguistic sophistication, in particular to deal with word order, agreement and subcategorisation constraints.

Table 8 Results of the evaluation of the sentence fusion output as the number of sentences in each of the three categories *perfect*, *acceptable* and *nonsense* per judge (J1 and J2), broken down in restatements, generalizations and specifications.

	Restate		Specific		General	
	J1	J2	J1	J2	J1	J2
Perfect:	109	104	28	22	89	86
Acceptable:	44	58	15	16	34	24
Nonsense:	41	32	19	24	54	67
Total:	194		62		177	

3.4 Discussion

Our early work on sentence fusion described here was only performed on a small corpus, consisting of parallel translations of a single book, (*Le Petit Prince*). We have not offered an evaluation of sentence fusion in the context of QA. Both limitations have been addressed in the DAESO project (short for Detecting And Exploiting Semantic Overlap), which was a partial continuation of the IMOGEN project. In this project, a one million word parallel monolingual treebank was developed (Marsi and Krahmer, 2007, 2009), containing multiple translations of various books (*Le Petit Prince*, but also parts of Darwin's *Origin of Species* and Montaigne's *Essays*), as well as multiple news reports about the same event, headlines for related news reports and multiple answers to the same question (this data was actually collected in the IMIX project).

Half of the data was manually annotated, for which two annotation tools were developed (both publicly available): Hitaext, allowing for many-to-many alignments on the sentence level, and Algraeph, for sentence alignment at the word and phrase level.[4] The other half of the corpus was automatically aligned, with a vastly improved version of the automatic aligner described above. One important aspect of the automatic aligner is that it makes no prior assumptions about the relation between the two sentences other than that they are somehow related: the amount of overlap may range from a few words (as may be the case in the news reports) to the entire sentence (as is the case in the parallel translations), and no order between the sentences is assumed. Evaluation (on alignment of news reports) shows that the performance of the automatic aligner approaches that of the human annotators, where it is interesting to observe that the algorithm outperforms human annotators on the Equals and Intersects relations. Human annotators, by contrast, are better at classifying the remaining relations, but since Equals and Intersects are relatively frequent in the News segment, the overall weighted performance of our algorithm is less than a percent below the scores obtained by the human annotators (Marsi and Krahmer, 2010).

In DAESO we also continued our work on sentence fusion. It has been argued that sentence fusion is a poorly defined task, which is therefore difficult to evaluate

[4] Both tools are available from `http://daeso.uvt.nl`

(Daumé III and Marcu, 2004). Krahmer et al. (2008) studied sentence fusion in a QA setting, where participants were asked to merge potential answer sentences with and without explicitly showing the question. We found that question-based sentence fusion is a better defined task than generic sentence fusion (Q-based fusions are shorter, display less variety in length, yield more identical results and have higher normalized Rouge scores). The sentence fusion data collected in this way has been made publicly available. Moreover, we found that in a QA application, participants strongly prefer Q-based fusions to generic ones, and have a preference for union fusions (combining information from answers, without overlap) over intersection fusions (only using the shared information in potential answer sentences). This clearly shows that sentence fusion is indeed a useful strategy for QA systems.

4 Conclusion

In this chapter we described two related ways in which QA systems could provide more informative answers, by (1) doing query-based summarization and (2) fusing potential answer sentences to more complete answers.

Concerning query-based summarization, our primary aim was to bring automatic content selection practice in line with insights from discourse theory. To this end, we devised a framework for automatic summarization which is founded on graph theory and can be applied as a text-to-text generation technique in question answering. The content selection algorithm is entirely based on relations between content units (text passages). The evaluated systems are just examples of possible implementations of this framework; they can be extended to exploit more textual features, and discourse oriented features in particular.

The framework represents a step toward context aware summarization. Previous work on query-based summarization has mainly focused on extracting the set of sentences which best match the query, ignoring their broader context. The features used for relating sentences are computationally low-cost and easy to port to other languages, but knowledge-intensive methods may detect relations between sentences more accurately. Despite this, the graph-based approach showed good results compared to DUC participant systems (the redundancy-aware probabilistic relevance system would have ranked first for Rouge-2 and second for Rouge-SU4 if it had participated in DUC 2005, when most of the work described in this chapter was done), which indicates that we are on the right track. The main lessons learned from our experiments are the following:

1. The graph-based approach to summarization represents a promising direction, given the good results in spite of the superficial linguistic analysis performed by the evaluated systems. Even better results are to be expected when more sophisticated features are used.
2. The probabilistic interpretation of semantic networks (i.e., *probabilistic relevance*) seems to be more suitable for content selection than the social network interpretation (i.e., *normalized centrality*).

Further performance gains may be achieved by using more different sources of information for detecting relations, including knowledge-intensive methods such as rhetorical relation detection or anaphora resolution.

This chapter also described our first explorations into sentence fusion for Dutch. Our starting point was the sentence fusion model proposed by Barzilay et al. (1999); Barzilay (2003), and further extended by Barzilay and McKeown (2005), in which dependency analyses of pairs of sentences are first aligned, after which the aligned parts (representing the common information) are fused. The resulting fused dependency tree is subsequently transfered into natural language. Our new contributions are primarily in two areas. First, we carried out an explicit evaluation of the alignment – both human and automatic alignment – whereas Barzilay (2003) only evaluates the output of the complete sentence fusion process. We found that annotators can reliably align phrases and assign relation labels to them, and that good results can be achieved with automatic alignment, certainly above an informed baseline, albeit still below human performance. Second, Barzilay and co-workers developed their sentence fusion model in the context of multi-document summarization, but arguably the approach could also be used for applications such as question answering or information extraction. This seems to call for a more refined version of sentence fusion, which has consequences for alignment, merging and realization. We have therefore introduced five different types of semantic relations between strings (i.e. equals, restates, specifies, generalizes and intersects). This increases the expressiveness of the representation, and supports generating restatements, generalizations and specifications. We described and evaluated our first results on sentence realization based on these refined alignments, with promising results.

References

K. Bakshi, D. Huynh, B. Katz, D.R. Karger, J. Lin, D. Quan, and V. Sinha. The role of context in question answering systems. In *CHI '03 extended abstracts on Human Factors in Computing Systems*, pages 1006–1007, New York, NY, USA, 2003.

R. Barzilay. *Information Fusion for Multidocument Summarization*. PhD thesis, Columbia University, 2003.

R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *Proceedings of the ACL workshop on Intelligent Scalable Text Summarization*, pages 10–17, August 1997.

R. Barzilay and K. McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328, 2005. ISSN 0891-2017.

R. Barzilay, K. McKeown, and M. Elhaded. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the ACL*, Maryland, 1999.

M.J. Bates. The berry-picking search: user interface design. In H. Thimbleby, editor, *User Interface Design*. Addison-Wesley, 1990.

S. Blair-Goldensohn and K. McKeown. Integrating rhetorical-semantic relation models for query-focused summarization. In *Proceedings of the Document Understanding Conference*, 2006.

G. Bouma, G. van Noord, and R. Malouf. Alpino: Wide-coverage computational analysis of Dutch. In *Proceedings of CLIN*, 2001.

J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, New York, NY, USA, 1998.

J. Carletta. Assessing agreement on classification tasks: the kappa statistic. *Compututational Linguistics*, 22(2):249–254, 1996. ISSN 0891-2017.

H. Daumé III and D. Marcu. Generic sentence fusion is an ill-dened summarization task. In *Proceedings of the ACL workshop: Text Summarization Branches Out*, Barcelona, Spain, 2004.

H. P. Edmundson. New methods in automatic extracting. *Journal of the ACM*, 16 (2):264–285, April 1969.

G. Erkan and D.R. Radev. LexRank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 2004.

D. Gildea. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st annual meeting of the ACL*, Sapporo, Japan, 2003.

E. Krahmer, E. Marsi, and P. van Pelt. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of the 46th Annual Meeting of the ACL*, pages 193–196, Columbus, OH, USA, 2008.

I. Langkilde and K. Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th annual meeting of the ACL*, pages 704–710, Morristown, NJ, USA, 1998.

C.-Y. Lin. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the ACL workshop: Text Summarization Branches Out*, Barcelona, Spain, 2004.

H.P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

I. Mani and E. Bloedorn. Multi-document summarization by graph search and matching. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 622–628, 1997.

W.C. Mann and S.A. Thompson. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8:243–281, 1988.

D. Marcu. Discourse trees are good indicators of importance in text. In I. Mani and M. Maybury, editors, *Advances in Automatic Text Summarization*, pages 123–136. MIT Press, 1999.

E. Marsi and E. Krahmer. Classification of semantic relations by humans and machines. In *Proceedings of the ACL workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 1–6, Ann Arbor, Michigan, 2005a.

E. Marsi and E. Krahmer. Explorations in sentence fusion. In *Proceedings of the 10th European workshop on Natural Language Generation*, Aberdeen, UK, 2005b.

E. Marsi and E. Krahmer. Annotating a parallel monolingual treebank with semantic similarity relations. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories*, pages 85–96, Bergen, Norway, 2007.

E. Marsi and E. Krahmer. Detecting semantic overlap: A parallel monolingual treebank for dutch. In *Proceedings of CLIN*, 2009.

E. Marsi and E. Krahmer. Automatic analysis of semantic similarity in comparable text through syntactic tree matching. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 752–760, Beijing, China, August 2010.

M. Maybury. *New Directions in Question Answering*. AAAI Press, 2004.

A. Meyers, R. Yangarber, and R. Grisham. Alignment of shared forests for bilingual corpora. In *Proceedings of 16th International Conference on Computational Linguistics*, pages 460–465, Copenhagen, Denmark, 1996.

E. W. Noreen. *Computer intensive methods for testing hypotheses: an introduction*. Wiley, New York, NY, USA, 1989. ISBN 978-0-471-61136-3.

F. J. Och and H. Ney. Statistical machine translation. In *EAMT Workshop*, pages 39–46, Ljubljana, Slovenia, 2000.

M.F. Porter. Snowball: A language for stemming algorithms, 2001. http://snowball.tartarus.org/texts/introduction.html.

K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

T. Strzalkowski, R. Gaizauskas, E.M. Voorhees, S. Harabagiu, R. Weishedel, D. Israel, C. Jacquemin, C. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, J. Burger, A. Singhal, C. Cardie, R. Shrihari, and V. Chaudhri. Issues, tasks, and program structures to roadmap research in question & answering (Q&A). NIST, October 2000.

T. van der Wouden, H. Hoekstra, M. Moortgat, B. Renmans, and I. Schuurman. Syntactic analysis in the spoken dutch corpus. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 768–773, Las Palmas, Spain, 2002.

P. Vossen, editor. *EuroWordNet: a multilingual database with lexical semantic networks*. Kluwer Academic Publishers, Norwell, MA, USA, 1998. ISBN 0-7923-5295-5.

F. Wolf and E. Gibson. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–288, 2005. ISSN 0891-2017.