MSc in Speech and Language Processing Dissertation : Automatic summarising based on sentence extraction: A statistical approach

Byron Georgantopoulos
Department of Linguistics
University of Edinburgh

October 1, 1996

Abstract

The present dissertation and project describes a system for automatic summarising of texts. Instead of generating abstracts, a hard NLP task of questionable effectiveness, the system tries to identify the most important sentences of the original text, thus producing an *extract*.

The proposed, corpus-based and statistical approach exploits several heuristics to determine the summary-worthiness of sentences. It actually uses statistical appearances of words, word-pairs and noun phrases to calculate sentence weights and then extract the highest scoring sentences. The statistical model used in the scoring function is a slight variation of the Term-Frequency Inverse-Document-Frequency (TFIDF) term weighting formula.

The results obtained by application of the system to 5 test texts, separately and cumulatively for the three afore-mentioned heuristics were subjectively judged for their quality. This evaluation showed that the noun phrases, as a separate method, produce the best extracts and that they also are parts of the best performing combinations.

Acknowledgements

I would like to thank my supervisors, Andrei Mikheev and Jim Hurford. Andrei was very helpful in the technical-programming issues as well as the general statistical NLP techniques. Jim provided the “linguistic” viewpoint and clever suggestions.

I would also like to thank the Greek State Scholarships Foundation, for providing the funding, without which I would not have been able to do this MSc.

Contents

Chapter 1

Introduction

The number of papers that are published today is ever increasing, while the Internet burst has created vast libraries of machine-readable texts. For one to keep informed and updated about the recent advances in his field has become now both vital and difficult. Since it is impossible to read through all the papers which are published nowadays, it is of great help to present them in a condensed way, i.e. using abstracts which summarise the content of an article. In this way the reader can get quickly a general idea about the article and decide if it is interesting enough to read it all.

However, the work of abstracting a document is far from being easy. It requires skilled and specialised abstractors and of course it takes plenty of time. Only after an abstract is prepared, can the paper be published, which results to an unavoidable and irritating delay. In the same way as machine translation, linguists and computer scientists have tried in recent years to substitute and/or aid human with machine abstractors. Automatic (or computer-based) abstracting, which started back in the late fifties, has shown considerable progress, has created several techniques and theories and has even produced some commercial software.

This thesis focuses on the production of a special kind of abstracts, called *extracts*: sets of sentences taken from the original text. These sentences are selected on the basis of the amount of information they carry about the subject content. Using statistical tools, the system "learns" from a corpus of papers which elements of a sentence make it important enough to function as a highly representative sentence. There are several such elements explored in the existing bibliography; the system utilises three of them: words, word pairs and noun phrases, both as separate units and cumulatively, (by combining two or all of them). It finally attempts to find the best combination, i.e. the one which produces the most satisfying set of extracted sentences.

1.1 Abstracts: Nature and characteristics

There is no generally agreed definition for abstracts although several efforts have been made to grasp every possible interpretation. Without trying to be perfectly exact, we can say that an abstract is a condensed representation of the content of a text. An abstract serves as a representative of a text[1], but much reduced in size. However, this reduction should entail minimal loss in content.

A successful abstract must have certain attributes. It must be **brief** in style and size, **accurate** (not under- or over- informative) and have **clarity** in expression. These virtues have been well documented[2] and require educated professors to apply them. A similarity can be drawn with the translators: both have to transform a given text, trying to keep faithful to it at the same time.

1.2 Uses of abstracts

Abstracts support the transfer of knowledge and communication within the academic and scientific community. They "inherit" all the purposes of paper publishing, but it is for their brevity that they:

1. Can keep the reader aware of virtually all the new developments and findings in his area
2. Reduce the time spent on an article and thus increase the number of papers of which one can have an idea of their basic conclusions
3. Help select the articles which are worth full reading
4. Require of the reader to know only the language used in the abstract journal, although the original paper could be written in any language
5. Facilitate literature reviews because it is easier to browse through abstracts than whole texts
6. Speed up document classifying by using abstracts as the input to the classification procedure

1.3 Types of abstracts

There are basically two types of abstracts: the **Indicative abstract** which gives a brief outline of the basic concepts and conclusions of the paper and

[1] As a surrogate, an abstract is placed between the title and the table of contents, in a size axis.

[2] There are numerous guidelines for the creation of abstracts. See (Borko & Bernier 1975) references for some Institution's published standards.

the **Informative abstract** which is usually longer and includes results, figures etc.

Depending on the author, abstracts can be **author prepared**, **subject-expert prepared** or **professionals prepared**. Author prepared abstracts suffer a criticism for bias and subjectivity; experts are probably the best choice, while professionals are competent but will not produce high quality work if they have little knowledge of the subject matter.

A final distinction is made between **human-produced** and **computer-produced** summaries. In the rest of the dissertation by *summary* I will refer to summaries (or abstracts or any other form of compression) made by humans.

Chapter 2

Automatic abstracting

2.1 Introduction

As Machine Translation started to show very promising results in the fifties, scientists felt confident to put their hands to every other NLP problem, including of course automatic abstracting. With the cumulating of vast amounts of texts it was of great help to condense published papers via computer programs. (Luhn 1959) created the first abstracting system based on word-frequency statistics. Nowadays, depending on the processing tools, on the available corpus and the research motives, one can choose between pure NLP, statistical NLP or hybrid methods. In order to select a method we need to know:

- What **kind of texts** we want to summarise. Domain, size and complexity of the texts affect the character of the system. For example, texts of a restricted domain are easier to summarise than multi-topical texts, because it might be feasible for the former to find and codify the rules-templates which produce the abstracts. Additionally, short papers call for different techniques in abstracting (and even in structuring and writing them in the first place) from dissertations and bigger articles.

- What **tools** we have and how self-confident we are about their performance. If we want to generate a coherent and comprehensible abstract we need an efficient text generation module. If we want to use statistics, we need a big and homogeneously annotated corpus along with corpus analysis tools.

2.2 Automatic abstracting through text understanding

This method is very close to the way humans do abstracting: the system has first to understand the text and then to create the abstract. The first step produces a canonical-logical form of the text that feeds the sentence generator to write the abstract. Both of these issues, text analysis and understanding[1], and text generation are huge NLP problems by themselves, and computationally unachievable so far. The complexities of language (anaphora, context, polysemy, global common knowledge required, etc.) make it too hard to create a system which will be effective in terms of producing good summaries, as well as being fast and maintainable[2].

2.2.1 Techniques

Methodologies for NLU-based summarising are borrowed mainly from information extraction systems. These systems have been tried to be tailored to automatic summarising for specific domains. They try to match their stored knowledge entities against the input text so as to locate its dominant concepts. The major drawback is that all these entities have to be hand-coded by humans. Usually the knowledge base has a huge size because it has to cater for every possible situation. Humans also undertake the maintenance and updating of the system.

2.3 Automatic extraction of sentences

Automatic summarising via sentence extraction has the target of locating the best content-bearing sentences in a text. Extracting sentences can be a much simpler and hence a fast and feasible approach. On the other hand the result passage might not be much comprehensible and refined: we sacrifice coherence for speed and feasibility.

The assumption behind extracting is that there should be a set of sentences which present all the key ideas of the text, or at least a great number of these ideas[3]. The goal is first to identify what really influences the significance of a sentence, what makes it important. The next step is to train

[1] Which might involve syntactic, semantic and discourse analysis.

[2] Systems built on a restricted domain could possibly show interesting results. See (McKeown & Radev 1995) for a project on summarising news articles.

[3] It is worth noticing here that many readers usually underline, emphasise with a marker, or circle important sentences or phrases, to facilitate a quick review afterwards or in the future. Others may read only the first sentence of some paragraphs to get briefly an idea of what the paper is about or just look for key words/phrases (also called scan or *speed* reading).

and program a system to automatically locate these elements in a sentence and compute its summary-value.

It is evident that this method avoids all the above-mentioned conventional NLP problems: no analysis, representation an understanding of the text is required. No generation has to take place, and in addition the extracted sentences will be perfectly grammatical.

2.3.1 Extracts vs. Abstracts

Although it is logical to suppose that the original text will include sentences that could be as good as the abstract sentences, there are several issues to be dealt with:

1. The sentences of an abstract are more dense and contain implications, generalisations and conclusions which might not be "expressed" intact in sentences of the main text.

2. The language style (especially in syntax) of an abstract is generally different from the original text[4]. Although an extract preserves the style of the writer, an abstract has a usually a more dense and conventional style.

3. The extracted sentences might not be textually coherent and might not flow naturally. It is much possible that there will be fragmentary sentences, which will not make sense in the context of the extract, in spite of being important sentences. Furthermore, the extract will probably contain unresolved anaphora[5].

4. There is a chance of redundancy and repetition in an extract, because sentences with similar content will both achieve high scores and will be extracted.

2.4 Techniques for automatic extraction

2.4.1 Rule-based approach

This method uses certain rules to select the sentences for the extract. The sentences which satisfy these rules are the ones to be extracted. Examples of rules are:

- Extract every first sentence in a paragraph

[4] Usually if the abstractor is not the original writer.

[5] Although the extract can be fed with more, originally unselected, sentences to make it look more like a normal text.

- Extract every sentence that has a title word(s) and a cue phrase (e.g. *This paper*)
- Extract every sentence with a specified number of words in a list. This word list should contain domain oriented words which are supposed to trigger worthy sentences.

The format of rules is usually an *if ... then* template:

```
IF (S(I) IS paragraph-initial) OR
   (S(I) CONTAINS ``This paper presents'')
THEN
   extract(S(I))
```

The two problems for NLP-based strategies referred above hold for this approach as well. The system must be provided with the rules by a human and these rules are specifically tailored to the domain they are written for. A change of domain should mean a major rewriting of the rules.

2.4.2 Statistical approach

In contrast to the human-provided rules, the statistical approach basically tries to learn automatically the rules which predict a summary-worthy sentence.

Statistics-based systems are empirical, retrainable systems which minimise human effort. Their goal is to identify the units in a sentence which influence its importance and try to learn the dependency between the significance of units and the significance of a sentence. In other words, what unit triggers a good sentence and in what quantities it is needed to do so; the occurrence of this unit will help to determine the weight of the sentence. In this framework each sentence is assigned a *score* that represents the degree of appropriateness for inclusion in a summary.

Statistical techniques for automatic extracting are very similar to the ones used in information retrieval. In the latter, each document is viewed as a collection of indices (usually words) and every index has a weight which corresponds to its statistical appearance in the document. The document then is represented by a vector with index weights as elements. In a typical document retrieval query[6] only the top frequency indices will be returned. In the same way, for extracting, each document is a collection of weighted sentences and the highest scoring ones comprise the final extract.

[6] As in the Search Engines for the World Wide Web, where the user queries all the existing documents for certain words.

Pros and Cons

The statistical methodology has two major advantages over NLP-based abstracting and rule-based extracting: For a crude system (only words) there is no need for language processing since all the scoring is based on superficial factors, i.e. the strings in a sentence. Furthermore, as it happens with statistical approaches, the same system can be used in different domains, as long as it can learn the statistics and the dependencies (be *trained*) on it before.

Statistics do not contradict the rule-based extracting. It is possible that they will come up with the same rules and factors for predicting the significance of sentences. However, statistically based systems learn these rules automatically from a corpus, and thus avoid the subjectivity and retrainability problems imposed by human coding. They also produce more refined rules because they are based on (combinations of) several predicting (diagnostic) units.

On the other hand statistical systems usually presuppose the independence of instances of the same diagnostic unit. This oversimplified model does not apply to natural language: words, noun phrases, etc. in a sentence do not appear completely independently. In addition, the selection of a statistical model which is responsible for utilising the statistical regularities of the existing corpus has a major effect on the performance of the system. Although a factor can be really important for summarising, an inadequate or irrelevant model may not exploit that. Hence, researchers usually attempt to find the best combination of factors as well as the statistical modelling which makes the best out of them.

Chapter 3

Outline of standard statistical NLP techniques

In this chapter are presented the tools, laws and techniques which are generally used in statistical NLP and in the implemented system.

3.1 Scoring the sentences

In out approach, the score of a sentence depends on its *diagnostic units*. A diagnostic unit (DU) is anything in a sentence which gives a clue to its significance. It can be a word, a sequence of words, a syntactic structure, its position within the text, etc. The presence and the importance of the DUs of a sentence virtually determines its score. The scoring formula can encapsulate more than one DU, for example, the goodness of a sentence can rely both on its terms and its position. The form of the formula follows the term-weighting model:

$$score = \sum_{i=1}^{n} a_{w_i} h_i$$

where a_{w_i} is the weight of the i-th heuristic, h_i is the specific score of the sentence for the i-th heuristic and n is of course the number of heuristics used.

The optimum weight of each contributing DU is found empirically, after several attempts with different weights. Ideally, the best combination is the one which maximises the score of only the extract-worthy sentences. In order to train to system for this, we need to know these sentences beforehand.

3.1.1 Determining weights

The main measure of a diagnostic unit (*term* in the following) weight is its statistical appearance in the specific document and its statistical appearance in the whole corpus. Statistical appearance can either mean how many times it is found in a collection (frequency) or if it appears at all (binary feature).

The assignment of weights must discriminate between relevant and non-relevant diagnostic units. There are several ways to calculate these weights (Robertson & Jones 1994):

collection frequency: if n is the number of documents term t_i appears in and N is the total number of documents then:

$$CFW_i = logN - logn = log\frac{N}{n_i}.$$

This method selects terms which occur often in a certain document but rarely in the rest by using the inverse of the term appearance.

term frequency: It takes into account only the frequency of a term inside the document:

$$TF_{ij} = \text{number of occurrences of term } t_i \text{ in document } j.$$

document length: It is logical to assume that terms appear more frequently in bigger files, so if a term is relatively more frequent in a short than in a big file, then it is more important. To incorporate document length in the weighting formula we define:

$$DL_j = \text{total number of term occurrences}^1 \text{ in } document_j.$$

This can be generalised to the average length of a document:

$$NDL_j = \frac{DL_j}{\text{Average DL for all documents}}$$

combined evidence: Richard & Jones combined together the three weighting formulas:

$$CW_{ij} = [CFW_i * TF_{ij} * (K+1)]/[K * NDL_j + TF_{ij}]$$

where K is a tuning constant which is determined after successive trials and adjusts the effect of TF_{ij}. K=0 eliminates the effect. TREC (Text REtrieval Conference) Program trials showed that K=2 is actually an effective value.

[1] This is not the same as the total number of words in a document because functional words do not count as term occurrences.

Term-Frequency Inverse-Document-Frequency (TFIDF): Another standard weight computation method (e.g. (Salton, Allan & Buckley 1993)) combines term frequency, the number of documents (N) and the the number of documents (n_i) that the term appears in:

$$weight_i = TF_i * log\frac{N}{n_i}$$

The TFIDF scoring formula also favours terms which are highly frequent in a document but rare in the rest of the corpus.

Frequency threshold

Making reliable inferences on occurrences counted in texts is not an easy task. Assuming that occurrences have a normal distribution, we will find many entries with very low frequencies. Are these entries good predictors? The answer is that it depends on the nature of the diagnostic unit. For single words, it is important to keep even the ones with very low occurrences because rare words are valuable[2]. But for word pairs, a different approach is needed. It is evident that words with low frequency will construct word pairs with low frequency as well. Since these rare words are already encapsulated and evaluated in the single-word method, they have no value in word pairs as predictors. Good word pair predictors rely on the mutual information between words. Common statistical practice[3] says that if the frequency of the term in question is below 5 then the statistical inference based on this frequency is unreliable.

3.2 Evaluation

Evaluation in NLP systems is a very difficult task; researchers have just recently began to define and systematise it. For the purposes of this project evaluation techniques for automatic information retrieval along with some specific techniques for evaluating abstracts and extracts are presented.

3.2.1 General Statistical NLP evaluation

Again measures of success are borrowed from automatic text indexing and classification[4]. Given that we have (perfect) extracts to compare against the computer-produced ones, the classical *precision* and *recall* ratios can be used. Their formal definition is:

[2] The above formulas have taken into account this fact by utilising inverse term frequencies.

[3] see also (Dunning 1993) for a more thorough explanation of statistics on frequencies.

[4] An introduction to Statistical NLP evaluation can also be found in (Finch & Moens 1995), a system for assigning descriptors in text.

$$\text{precision} = \frac{\text{number of correct items retrieved}}{\text{number of correct items}}$$

$$\text{recall} = \frac{\text{number of correct items retrieved}}{\text{number of retrieved items}}$$

For example, suppose there is a paper for summarising, and a set of sentences T (represented by their indices for convenience) have been identified before by a human. Then the system is tried on the same paper and produces the set C.

If T comprises of {S1, S2, S3, S4, S5, S6} and C comprises of {S1, S7, S4, S8, S9} then[5]:

	S1	S2	S3	S4	S5	S6	S7	S8	S9
human	√	√	√	√	√	√			
computer	√			√			√	√	√

Using the formulas, recall is two correct sentences out of five (40%), and precision is two correct sentences out of six (33%).

Precision indicates the ratio of correct hits over the total selected sentences. Recall indicates the ratio of similarity between the correct sentences extracted and the sentences in the human, "correct" extract (usually called the *target extract*). It is obvious that both quantities cannot exceed 100%.

Precision and recall are inversely related. If more sentences are extracted, recall will grow up as a result of more correct sentences being retrieved but precision will decrease because not all of the extra sentences will be correct. A combined measure, F-score (see Hobbs *et al.* (1992)) can be used:

$$F = \frac{(\beta^2+1)PR}{\beta^2 P+R}$$

where P is precision, R is recall and β is an parameter indicating the relative weight of precision and recall. If $\beta = 1$ there is no preference, if $\beta >$ 1 recall is more important, if $\beta < 1$ it is precision that is more favoured.

There are certain assumptions behind these measures; (1) all sentences carry the same weight, so whichever omission has the same cost, and (2) that there is a unique, ideal target extract. In fact, (1) some sentences are more important than others in the construction of an extract, and (2) if a sentence (or more generally a collection of sentences) does not belong to the target extract, can nonetheless be significant for the subject content and act as surrogate (without great loss) of the target sentence.

[5] the indices are arbitrary, they do not correspond to the place of the sentence in the body text.

3.2.2 Specific factors for automatic summarising

In addition to the afore-mentioned evaluations, there are some qualitative criteria specifically for abstracts:

- **Readability**: The text is understandable and reads easily.
- **Textual coherence**: There are no discourse gaps between the sentences and the text flows naturally.
- **Accuracy**: All the important issues of the text are covered and there are no errors.

3.2.3 Evaluating extracts

For the present system, the interest lies in (1) how informative is the extract compared to the human produced abstract/extract and (2) which heuristic or combination of heuristics performs best.

Usually the success of a method is measured against some baseline techniques which produce an extract without any real processing. One such method is the random selection of sentences (also used in (Edmundson 1969)). Other methods, slightly more advanced, include: (1) selecting sentences only from the beginning and the end and (2) just consisting of whole sections (e.g. the summary section).

Target extracts

Comparing against a preselected set of sentences means of evaluating how good the computer "simulates" humans in pinpointing the important sentences in a text. The advantage of this comparison is that it evaluates computer extracts as extracts and not as abstracts. Virtues beyond pure extracting, such as normal flow of ideas and resolved anaphora, are not compulsory.

The creation of target extracts (also called *manual extracts*) is made by human evaluators. They are usually required to read through the text and then mark the most important sentences. If the summarising system has no way of dealing with text coherence, there is no need for the target extract to have that attribute. Otherwise the human extractor has to pick up sentences which produce a connected and coherent passage.

Text similarity

Besides precision and recall measures we can also compute the mathematic similarity between computer and target extracts. This can help to demonstrate how close are the two passages. Using the *vector space model* (see (Salton 1989)) we represent the extract of a document of N sentences as a vector:

$$E = (s_1, s_2, ..., s_N)$$

where s_i has a value of 1 if the sentence belongs to the extract or 0 otherwise. If E_h is the human produced target extract and E_c is the automatic extract their similarity is calculated by multiplying the vectors:

$$sim(E_h, E_c) = E_h * E_c = \sum_{i=1}^{N} s_{h_i} s_{c_i}$$

where s_{h_i} and s_{c_i} are the elements of the vectors that represent the target and computer extracts respectively.

Sentence similarity

Measures based on the binary notion of belonging/not belonging to the target extract certainly ignore the possibility that there may be a surrogate subset of sentences which is satisfactorily close to the target extract. For example consider these two sentences taken from the same text (from the introduction and the summary respectively):

1. This paper describes the adaptation work carried out to retarget the Xerox Tagger to Spanish.
2. This paper presents results obtained with the port to Spanish of a public domain tagger – the Xerox Tagger.

It is obvious that these sentences have about the same content, but if one sentence makes it to the extract and then the program chooses the other, the cost for evaluation is the same as if any other sentence was selected. If we could find a way to express that two sentences are "similar" in content then the evaluation of an extract could be more precise. There is even a chance of judging the extract against a real abstract if a target extract is not available.

Sentence similarity can be influenced by morphological, syntactic and semantic criteria. Sentences with a great number of identical word stems or noun/verb phrases or similar logical forms can be said to be similar and thus interchangeable for inclusion in an extract.

Subjective evaluation

For a more integrated study of evaluation, it is convenient to judge the extract, as a whole and its sentences, as a stand-alone text excerpt and not against other targets. This method is based on the qualitative and quantitative evaluations of a human reader, who assigns scores to each computer-produced sentence on a point scale (usually consulting the original text and probably the abstract). It is the most unbiased method because the evaluator can value every sentence without being preoccupied by a target extract, and so an alternative extract can be equally appreciated if it is informative enough.

Chapter 4

Methods and Heuristics in automatic extracting

4.1 Review of extracting systems

Five systems on automatic extracting are reviewed to give a general idea of the historical background and progress. Edmundson is the basis, Kupiec is the most recent one, NetSumm is a commercial product and Salton explores a somewhat different methodology, but with interesting thoughts. Finally, Ono incorporates text understanding while avoiding using vast knowledge bases.

Note: Words in italics are terms used throughout the specific paper.

4.1.1 H.P. Edmundson's Abstract Generation System

It is not so risky to say that Edmundson's work (Edmundson 1969) set the trend for automatic extracting; all subsequent researchers refer to his work, and his heuristics have not been seriously expanded since then.

At that time, the only previous work on automatic extracting was Luhn's (Luhn 1959) system, ten years before, which only used high frequency words to calculate the sentence weights. Edmundson described and utilised cue phrases, titles and locational heuristics, tried all of their combinations and evaluated the computer-produced extracts with human-produced target extracts.

Preparation of target extracts

Edmundson carefully outlined the principles of human extracting in order to make his evaluation more concise. For a sentence to be *eligible* for the target extract it was required to carry information of at least one of six types: *subject matter, purpose, methods, conclusions or findings, generalisations or implications* and *recommendations or suggestions*. The final set of selected sentences must also be coherent and their number should equal 25% of the original text. Interesting note: headings were also included in these extracts.

Utilised heuristics

The heuristic (*clues*) come from two separate sources: the *body* and the *skeleton* of the text. They also can be *linguistic* or *structural*. All the possible combinations create four different *methods*:

		Structural Sources	
		Body	Skeleton
Linguistic Sources	General Characteristics	CUE METHOD	LOCATION METHOD
	Specific Characteristics	KEY METHOD	TITLE METHOD

Rationale of the four Edmundson methods

- **Cue method**: Basically cue phrases and words (see section 4.2.3) A Cue dictionary was constructed, including three different kinds of entries: *bonus words* (positive effect), *stigma words* (negative effect) and *null words* (the equivalent of stoplist words). Linguistic analysis showed which grammatical categories belong to these classes.

- **Key method**: Again, a Key glossary is created, which contains words with a corpus frequency above a certain percentage of the total number of words.

- **Title method**: It assigns positive weights to words appearing in titles, headings and formatted text.

- **Location method**: It assigns positive weights to headings, sentences in first and last paragraphs and paragraph-initial and paragraph-final sentences as well .

All these methods were tried singly and cumulatively and the best combination was chosen on the basis of the greater mean percentage of sentences common in the automatic extracts and the target extracts. The combination cue + title + location had the best mean co-selection measure (cue + title + location + frequency had the second best) and was later input to the next evaluation stage. The fact that the frequency heuristic did not improve the performance was a very important finding: (1) automatic extraction systems needed more sophisticated representations than single words, (2) the exclusion of word frequencies speeds up the system and reduces the storage requirements.

Evaluation studies

Two different evaluations (for the preferred method) were tried out:

1. **Subjective similarity ratings** measure the similarity between automatic-target extracts, based on subjective evaluation of similarity between their sentences (on a five-point scale). The result was a mean similarity of 66%.

2. **Statistical error analysis** measures the portion of the automatically extracted sentences worthy of belonging to a successful extract. Such sentences were specified as (1) identical with sentences in target extracts, (2) co-intensional and (3) extract-worthy judged. A promising 84% was classified in these categories.

4.1.2 Kupiec's Trainable Document Summariser

Kupiec *et al.* work (Kupiec, Pedersen & Chen 1995) is a highly influenced by Edmundson (Edmundson 1969)). Document extraction is viewed as a statistical classification problem, i.e. for every sentence, its score means the probability it can be included in a summary.

Heuristics

The heuristics (called *features* in the paper) utilised were:

- **Sentence length cut-off feature**: Sentences with fewer than 5 words are excluded.

- **Fixed-phrase feature**: Favours sentences containing any of 26 indicator (cue) phrases, e.g. "this letter...".

- **Paragraph feature**: Classifies the sentences into five categories: first 10, last 5, paragraph-initial, paragraph-final and paragraph-medial.

- **Thematic word feature**: All the high frequency words comprise a term vocabulary; sentences with such terms get extra points.

- **Uppercase word feature**: Favours sentences with proper names (in capitals).

The probability measure, using Baye's rule, that a sentence s will be included in the summary S is:

$$P(s \in S | F_1, F_2, ..., F_k) = \frac{P(F_1, F_2, ..., F_k | s \in S) P(s \in S))}{P(F_1, F_2, ..., F_k)}$$

and assuming statistical independence of the features, this reduces to:

$$P(s \in S|F_1, F_2, ..., F_k) = \frac{(\prod_{j=1}^{K} P(F_j|s \in S))P(s \in S)}{\prod_{i=1}^{K} P(F_i)}$$

$P(s \in S)$ is constant so it does not affect the probability, and the two remaining quantities can be calculated straightly by counting occurrences.

Evaluation

The project corpus consisted of 188 document/summary pairs; the summaries were manually generated by humans . 79.4% of the summary sentences were also found in the body of the texts (*direct match*). For another 3.3% there were two or more sentences in the texts which effectively had the same content (*direct join*). Therefore, the maximum precision rate for the system could reach the sum of these probabilities, 83%, since the rest of the sentences in the summaries did not have any direct or indirect correspondent in the original texts. Kupiec calculated recall and precision scores both individually for each feature and with combinations. The conclusion supported paragraph + fixed-phrase + sentence-length, with a 42% recall[1].

The notion of direct matches and joins is a useful one when target extracts are not available. Furthermore, the results are in trend with Edmundson; both studies show now a clear preference over certain features (location, cue phrases) and reduction of performance for the word-based statistics. However, we have to consider two facts which somehow decrease the significance of the results. First, that manual summaries had on average only three sentences. Second, that the fraction of direct sentence matches found in the corpus was surprisingly high. (for instance, in the Computational Linguistics corpus used to train the present project, rarely sentences in the abstract coincided with text sentences)

4.1.3 Salton's Passage Retrieval System

Salton *et al* (1993) system, *Smart*, does not produce straight abstracts, but tries to identify sets of sentences (even whole sections or paragraphs) which represent the subject content of a paper. In the report, there is a brief introduction to sentence extracting, and it is stated that retrieving passages is a right step towards better response to user queries. The architecture of Smart is based on vector processing model, i.e. every document is represented as a vector of weighted terms. The procedure to obtain the most relevant documents is as follows:

1. The user supplies a query: it may consist of a word, a stem or a phrase.
2. For every document a similarity metric is computed between the query and the document, based on the number and the weight of co-occuring

[1]There is only one figure for precision, 35%, available in the paper.

(in the query and in the document) terms. This is called the global similarity.

3. All the documents having a global similarity below a predefined threshold (normally it is 0.2) are rejected.

4. To overcome the problem of ambiguous words (with multiple meanings) and phrases the local text environments are considered[2]; only texts with a sufficient number of locally similar substructures finally qualify for the response to the query.

Results

The effectiveness of the system is enhanced by considering ways to retrieve subsets of texts with the same qualifications as the previous texts, i.e. with a similarity metric above a certain threshold. In this way, sentences, sequences of adjacent sentences, sections and paragraphs are assigned a similarity metric (using the above-mentioned stages) and the ones with a high similarity can be considered as surrogates for the original full texts.

The evaluation is based on recall and precision figures, along with statistics that show the ratio paragraph per retrieved item, indicative of the achieved compression. The utilisation of local similarity and afterwards the substitution of texts with text excerpts improves the efficiency of the system by 25%. Salton concludes that the best performing combination is sections+paragraphs, with averagely 1.3 paragraphs representing each document.

Although not a strictly sentence extracting system, Smart deals with issues (word ambiguity, similarity between sentences) in the area of automatic extracting. Considering sequences of sentences instead of isolated sentences also decreases the effect of problems such as anaphora resolution and textual coherence.

4.1.4 NetSumm

NetSumm is a product of British Telecommunications and summarises WWW Pages. It "automatically highlights the sentences that it considers make up the "most important" part of the text". BT has established a WEB page, (`http://www.labs.bt.com/netsumm/pages/about.htm`) to spread information about the program and it allows you to submit a paper and get the extract. Unfortunately, the methodology is not explained, so it remains a secret (obviously for commercial reasons) how the extracts (*abridgements*) are produced.

[2] It was observed that many non-relevant documents produced high scores because they contained the same word (or stem) but a different term the word had multiple senses. This can be avoided if the context of the sentences is taken into account.

NetSumm is program running in Mackintosh computers and through NetScape thus using all the user-friendly facilities, like mouse, menus and buttons. It accepts ASCII and HTML formatted documents and it responses very quickly. It allows for some parametrisation of the proportion of condensation, and it can display the extract sentences alone, or the whole text with the extracted sentences highlighted.

Evaluation

In accordance with (Morris, Kasper & Adams 1992) BT assert that "an abridgement of typically only 5% of the original article will contain roughly 70% of the information in an author-written abstract, while a 25% abridgement contains essentially all of the information in the abstract".

4.1.5 Abstract Generation based on rhetorical structure extraction

Ono *et al.* (1994) developed a system that generates abstracts of Japanese expository writings based on the notion of rhetorical structures. It is basically a pattern matching technique which tries to find the structure of mostly argumentative texts. It also ensures that the output extract will be coherent, whatever the desired length.

Rhetorical structures represent deep semantic relations between various portions of a text, in two layers: intra-paragraph (between sentences) and inter-paragraph. These structures consist of rhetorical relations. Typical rhetorical relations (and the connectives which reflect the flow of the argument) include serial (*X thus Y*), summarisation (*X after all Y*), reason (*X because Y*) etc.[3]. Rhetorical structures are stored in binary trees (in the form of *connective(X, Y)*, so there is a hierarchy of relations - the higher in the tree the more general and important the relation.

Rhetorical structure extraction

The extraction procedure works as follows: the sentence is morphologically and syntactically parsed and then the basic rhetorical relations are detected (like the ones referred above) between adjacent sentences . The segmentation module detects rhetorical relations between distant sentences and then the candidate generator produces all possible rhetorical structures. The final step is to assign penalty scores to each candidate. Penalty scores are based on preference rules for every neighboring relations (i.e. what is the most preferred structure for a given relation).

[3]There is a clear resemblance with cue phrases/words, explained below

Abstract Generation

The generation of the abstract is accomplished by examining the best performing rhetorical structure of each section. Rhetorical relations are categorised according to which operand is the stronger: for example in summarising relation *summarisation(X, Y)*, *Y*, the conclusion, is obviously more important than *X*. This relation is characterised as *RightNucleus* and there are two other types: *LeftNucleus* and *BothNucleus* relations.

The system evaluates the importance of all the sentences by imposing penalties to the weaker parts of the categorised relations. The nodes of the tree with the highest penalties (from bottom to top) are cut out and the remaining terminal nodes construct the extract[4].

Extracts of these type carry two important virtues: the relations and the connectives represent true relations and arguments of the original text. Secondly, they will not contain fragmentary sentences; all the sentences are there as parts of rhetorical relations. In addition it is easy to vary the length of an extract: just cut out more terminal nodes, the ones with the highest penalties.

Evaluation

The system was evaluated against 30 editorial articles and 42 technical papers. Recall figures were 51% and 74% respectively, and this difference was explained on the basis that technical papers have a more typical and expository style and contain more explicit clues of rhetorical relations.

4.2 Review of diagnostic units

In the following sections all the diagnostic units are reviewed in more detail.

4.2.1 High frequency keywords

The first and simplest method tries to utilise the basic units of a sentence: words. It assumes that the most frequent words found in the body of a text might be indicative of high content in the sentences they appear in. However, such a word must have another attribute: it must be infrequent in the rest of the corpus, because otherwise it is just a common word for the domain without any discriminating power[5]. For example, the word *linguistics* will possibly appear very often in a computational linguistics corpus. In a sentence of the text we want to summarise, this word does

[4] Inter-paragraph reduction and arrangement of the connectives also takes place.

[5] This is the reason why we need a large and homogeneous corpus: to be able to find what is the occurrence of a certain word in a collection of texts. The larger the corpus, the more correct the conclusions about a word's distribution.

not enhance its significance; anyway we expect a lot of sentences in such a corpus to contain the word *linguistics*. The same holds for the functional words in English, like *the*, *and* etc. They have very high frequencies and even distributions throughout the corpus, and their contribution to the content of the sentence is negligible.

It seems fairly logical that some (though certainly not all) the key ideas for a paper will repeat throughout the sentences, but it cannot be easily concluded that:

- every major concept will have a reasonably high frequency in the paper and a poor frequency in the rest of the corpus.
- these key concepts are just single words; they may extend over one word, e.g. noun phrases.

  and what's more:

- high frequency words which are not concepts but happen to appear a lot of times in a text will boost undesired sentences.
- words with multiple senses (e.g. *bank, book* or parts of bigger structures like *car oil* and *olive oil*) will be grouped into a single one and thus blur the terminology bank.

Word-based statistics are the easiest to be obtained because they do not require any advanced linguistic processing. This methodology was one of the first to be tried out ((Luhn 1959), (Rath, Resnick & Savage 1961), (Edmundson 1969)). One of its main advantages derives from a statistical law: the greater the collected statistics, the more reliable the method. Indeed, the highest frequencies will appear for this heuristic because all the other frequency based[6] diagnostic units consist of more than one word and their chances of repeating will be much less than of those of the single-word ones[7].

4.2.2 Word pairs

In the next level, these terms are taken to be pairs of words. The idea behind this concept is that the co-occurrence of two certain (not necessarily adjacent) words in the same sentences in high frequency could serve as a diagnostic unit[8]. First, terms which consist of two words can be grasped. Second, if a document contains relatively many sentences with two certain words, this must relate to its importance (either as a specific term or as typical of the writer's style) for this document. Third we are able to evaluate

[6] Locational heuristics are not of course affected by this.

[7] For example the frequency of *computational linguistics* in a collection of papers is surely less than each of the frequencies for *computational* and *linguistics* respectively.

[8] Word order is not considered as important, at least for the present system.

context information; if a word tends to appear with another one frequently their dependence will produce high frequency word pairs and thus be valued accordingly.

A further elaboration of the method could be to select which part of speeches are allowed to form a word pair candidate[9]. A problem with this approach derives from word ambiguity when considering adjacent words. For instance, consider the famous linguistic phrase: "Time flies like an arrow". The first three words here can belong to more than one grammatical categories so it is not clear which grammatical combination this construct represents. If it appears in a later sentence with different categories this time, then it will unfortunately group with the previous one.

4.2.3 Cue words and phrases

It is typical in writing a paper for publication to use certain words or phrases to highlight sentences. It was Edmundson (Edmundson 1969)) who first observed and used these heuristics for his project. Such cues as *This paper ...*, *we propose ...*, *important* are frequently indicative of a worthy sentence for summarisation. These diagnostic units are called *cue phrases* and *bonus* words. On the contrary, there are some negative words (or phrases), called *stigma* words which decrease the significance of a sentence, like *hardly*, *impossible*, belittling expressions, etc.

4.2.4 Title and Headings words

Undoubtedly the title and the headers[10] of a text provide a good amount of information about its content. By extracting the content words from titles and headers we can create a scoring formula which increases the score of sentences that contain these words. We can also put headers and titles directly into the extract thus creating a more coherent and structured output. Finally, the system can be guided to select at least one sentence from every major section, so given that this sentence is good, the extract is more likely to grasp all the basic subjects.

4.2.5 Utilisation of already existing abstracts

When abstracts (or even better extracts) are available with texts, the system can "learn" certain properties from them. We can extract words, noun phrases, syntax style, etc. from abstracts and then try to find similar sentences in the body of the text. In this way the output sentences bear more resemblance (in content and in format) with the human-produced abstracts.

[9] For example, we might be especially interested in pairs noun+noun.
[10] Except of course trivial headers: *Introduction*, *Conclusions*, etc.

Other statistical properties which can be utilised include the average number of sentences in a summary along with the average length of summary sentences.

4.2.6 Linguistic processing

Linguistic information can also prove useful, on the basis of looking for strings of words that form a syntactic structure. Extending the idea of high frequency words, we can assume that noun phrases form more meaningful concepts, thus getting closer to the idea of *terms*. This overcomes several problems of the first single-word method because it can utilise:

- compound nouns
- terms which consist of adjective+noun, e.g. *computational linguistics*

There is a possibility though that one term can be "implemented" with more than one noun phrases, compare for example *information overload* with *overload of information*.

4.2.7 Locational heuristics

The position of a sentence within a paragraph (and the position of a paragraph within a text as well) can sometimes be indicative of its significance. This happens usually with the first sentence of a paragraph which supposedly gives a very good clue of what is going to follow. Another similar assertion is that a good proportion of important information about a paper is to be found in the first and last paragraphs (they indeed often contain introductory and conclusive material), so their sentences should be favoured more than the sentences of the other paragraphs.

4.2.8 Exploiting text formatting

The format of an article can further aid the searching for summarising sentences. Writers usually put important phrases in bold or italic font, capitalise or underline them if they want to give a certain emphasis.

4.3 Additional factors affecting a system for automatic extraction

Along with the diagnostic units we can further parameterise the system and experiment with varying values in order to find optimum combinations.

4.3.1 Length of an extract

Morris *et al.* (Morris et al. 1992) postulated that about 20% of the sentences in a text could express all the basic ideas about it. Since abstracts are much shorter than this proportion, the length of extracts should lie between the length of an abstract and Morris's percentage[11].

There are five ways to vary the number of sentences for an extract:

1. **Proportion**: Select a predefined percentage (usually 10%) of the number of sentences of the paper. This technique is good for normally sized papers but will produce long extracts for long documents. Since long documents may contain many ideas and probably more than one topic it might be helpful to try for a lot of sentences. However, long extracts magnify the problems of the method (incoherence, repetition) and so many sentences can be confusing to read and follow. Finally, from an economical point of view, big extracts contradict the brevity of a summary: nobody would like to read a summary with 100 or more sentences.

2. **Oracle method**: If a target extract is available, select the same number of sentences. Precision and recall figures are always the same because of the same denominator. However, target extracts are not supposed to be accessible (otherwise why making a computer program?). In addition, it is intuitive that a computer extract will need more sentences than the perfect extract to reach a good point of coverage and coherence. But an advantage of the oracle method is that the system can be "trained" from the target extracts so it can predict for the test documents the optimum number of sentences.

3. **Fixed number of sentences**: Here the length of an extract is always the same (typical number: 10-15 sentences) regardless of the size of the paper. This technique is closer to human-produced abstracts. It favours shortness but the problems of the previous method are replicated: who knows how good the runner-up sentences would be?

4. **Sentences above a certain threshold**: For a sentence to be included in the extract it suffices to have a reasonable enough score. This is a way of compromising the extremes of the previous methods, but it requires to determine this threshold.

5. **Mathematic formula**: The number of extracted sentences is an increasing function of the number of sentences in the text, but it does not grow linearly. So, relatively few sentences are added for a big text

[11]The density of content of human abstracts inevitably puts a lower limit to the number of sentences an extract can contain

and even fewer for a much bigger. This is probably one of the best methods, it prevents a size explosion but it caters for huge papers as well[12].

4.3.2 Length of a sentence

It can be suggested that too short or too long sentences are not generally ideal for an abstract, and therefore for an extract as well. This is usually referred to (see (Kupiec et al. 1995)) as *sentence cut-off feature*. It penalises short (less than 5-6 words) and long sentences either by reducing their score, or even by excluding them completely[13].

4.3.3 Stop words

Language in general contains much redundancy. There are many words that carry no meaning but are necessary to construct grammatical sentences. These words include prepositions, pronouns, articles, connectives etc. and are called *function words*. It follows that the function words should not be included in any statistics and scoring formulas because they do not contribute to the relevance and importance of a sentence. For the present system I used an enhanced list that contains along with the traditionally functional words, some others as well which intuitively do not affect the summary-worthiness of a sentence, e.g. *following, first*, auxiliary verbs, etc[14].

4.3.4 Word stemming and Capitalisation

Can words of the same morphological root be considered identical terms? Especially in automatic indexing, it is a classical technique to use word stems as indices. It speeds up the procedure, reduces table sizes and supposedly enhances the performance of the system. However, there is no clear evidence that words with the same stem but with different suffixes (and possibly different grammatical categories) must be grouped. A difference in morphology sometimes entails a semantic difference, even a subtle one. Moreover, the word ambiguity problem is considerably enlarged as less different words exist. Finally, the quality of the stemming program has a strong impact on the usefulness of the technique. (Strzalkowski 1992) refers to an 6%-8% improvement of precision by incorporating morphological information. Similar assertions can be made about capitalised and non-capitalised words. Others distinguish between for example *Grammar* and *grammar*, others group them

[12]See (Borko & Bernier 1975), pp.170-171 for such a formula.

[13]The nature of the scoring formula which usually divides by the number of words in a sentence, ensures that long sentences are anyway not favoured since they will have a greater dividend.

[14]See Appendix 2 for a complete list of these words.

into one word. No approach is definitely better than the other, it is rather a matter of preference.

Chapter 5

Description of the Corpus

The nature and characteristics of the corpus have a strong impact on the later stages of work. It is of great importance to design beforehand the overall structure and appearance of the corpus, the ratio of automatic/manual editing required and adapt it to the annotation and processing tools which will be used afterwards.

The corpus processing was carried along with Simone Teuffel, PhD student in the Centre for Cognitive Science, who also did a summer project on automatic summarising.

5.1 Specifications

Statistical NLP requires a large amount of texts to guarantee the reliability of the gathered statistics. These texts must also be in electronic form and belong to the same domain, since we want to locate high content diagnostic units specifically for this domain. For the task in hand, at least the files that will later comprise the test set should contain an abstract (or even better an extract) to serve as a target for evaluation. Finally, they must be files of the same format, so the same processing programs can be applied to all the texts and output a uniform and coherent corpus.

After an extensive search in the World Wide Web (WWW), the only group of texts satisfying all these criteria was a collection of papers on Computational Linguistics, all of them LaTeX files. Although enormous amounts of texts, on several fields, are available online, they are only in the format of postscript files. Postscript files are by nature files to be displayed on a graphic screen, it is extremely difficult, if not impossible, to reconstruct them as text.

5.2 Origins and domain

The corpus consists of papers downloaded from the **Computation and Language E-Print Archive** (`http://xxx.lanl.gov/cmp-lg`) which contains papers submitted from June 1994 until May 1996. All the papers downloaded were LaTeX files and contained small abstracts. The topics included: computational linguistics, natural-language processing, speech processing, and related fields.

5.3 Corpus Processing

After the LaTeX files were downloaded they had to pass several processing stages in order to be usable by the Language Technology Group (LTG, University of Edinburgh) tools. LTG programs require the files to be in SGML format (see below).

5.3.1 The SGML environment

SGML (Standard Generalized Markup Language, (Goldfard 1990)) is an international standard language for describing marked-up electronic documents.

In this language we list and describe all the allowable *entities* in a text (e.g. titles, paragraphs, sentences, words etc.)[1]. The structuring of the entities is hierarchical and is represented in a tree; there is a root entity and all the others have to belong to a parental node.

The major advantage of SGML for our approach is that it allows for different representations of a text to be constructed in a uniform way. The software that runs on the corpus can easily access whatever part of the text, the only required information is the name of the entity which corresponds to a certain diagnostic unit. It does not need to know how it was made or where it is placed.

Every entity (also called *structure*, or *tag*) has a name and a description, stored in a description type document (DTD). This file is consulted when marking up documents[2]. For a tag `T` its start and end are denoted by `<T>` and `</T>` respectively. For example if we have a sentence:

```
This is a test sentence
```

and if only sentences (tag: `S`) and words (tag: `W`) are used for markup, the SGML equivalent is:

```
<S><W>This</W> <W>is</W> <W>a</W> <W>test</W> <W>sentence</W></S>
```

[1]In fact, SGML is so general that can encode anything from poems to database entries.
[2]DTD is another standard language.

The structure of the sentence is (assuming TEXT is the root element):

```
TEXT  →  S  →  W  (This)
               W  (is)
               W  (a)
               W  (test)
               W  (sentence)
```

If we would also like to annotate noun phrases, the sentence would look like this:

```
<S><W>This</W> <W>is</W> <NG><W>a</W> <W>test</W>
<W>sentence</W></NG></S>
```

It is obvious that every new representation can be easily incorporated in the same text. This SGML sentence contains now two different kinds of diagnostic units.

The specifications set by the DTD file must be strictly satisfied. Except the imposed hierarchy (for example a `<W>` must appear only enclosed in an `<S>`, tags of the same category cannot be nested (e.g. sentences within sentences are not permitted).

5.3.2 The Language Technology (LT) library of Normalised SGML tools (NSL)

LT NSL is a development environment for SGML-based corpus and document processing, with support for multiple versions and multiple levels of annotation. It consists of a C-based API for accessing and manipulating SGML documents and an integrated set of SGML tools[3].

LT NSL belongs to a greater group of tools, the LTG corpus workbench[4], which also includes tools for tagging, tokenizing, frequency counting, etc.

The LTG tools work with SGML queries, which are requests for specific tags inside the documents. After the tag is retrieved it can further be processed.

5.3.3 Pre-editing the LaTeX files

The complexity of LaTeX files and the diversity of environments that different writers use had to be reduced before feeding the next module. We undertook some automatic deletions and alterations so as the files could be more easily and quickly converted to HTML.

[3] It was (and being) developed in the Human Communications Research Centre (HCRC) in the University of Edinburgh.

[4] For more information on LTG software see (Thompson & McKelvie 1996) or look at the LTG Internet site `http://www.ltg.ed.ac.uk/software.html`.

5.3.4 From LaTeX to HTML

Nikos Drakos' (Computer Based Learning Unit, University of Leeds) LaTeX2HTML program was used to convert the LaTeX files to HTML format[5] LaTeX2HTML is a powerful program, that basically tries to translate as many Latex commands as possible, including figures, equations and even user defined environments[6]. Since the summarisation program does not need this type of information, the converter is configured only for text. The command is :

```
latex2html -info 0 -nolatex -no_images <LaTeX_file>
```

The options before the filename tell LaTeX2HTML to avoid certain conversions of LaTeX commands.

Necessary replacements

Several LaTeX-oriented tags (like images and equations) whose presence and content do not have a summarising value were replaced by simple indicator tags. In this way the system did not count them as diagnostic units[7] but the reader of the text was able to know where an equation or an image was originally placed, although the actual content of the equation is not of course reproduced. These tags were:

1. **Images, figures**: It is obvious that they are not needed, and of course they cannot be represented in text mode.

2. **Equations**: A certain assumption can be made that an equation cannot be included in "high content" words, let alone that equations from a LaTeX file are not possible to represent in text mode. However one can argue that the presence of an equation can affect the significance of a sentence, especially for informative abstracts.

3. **References**: The names of references (authors and years) as well as references to other sections of the text are not indicative of sentence content.

[5] HTML stands for HyperText Mark-up Language, and is basically SGML with a certain DTD. It is widely used to create WWW documents.

[6] Although the submitted papers had to meet some requirements (they had to include title, abstract and author), writers tend to create new LaTeX commands of extreme diversity. LaTeX2HTML cannot cope with all these different commands and so the simplest version had to be used.

[7] They were inserted in the stoplist.

LaTeX file	Indicator Tag in HTML file
table	[#TABLE]
equation	[#EQN]
graphic image	[#IMG]
reference inside the text	[#CREF]
citation reference	[#REF]

Table of Indicator TAGS

5.3.5 Manual post-editing of HTML files

The diversity of papers necessitated some further editing of files. Among the most frequent corrections were:

- **Abstract and Title:** Keywords and Acknowledgements inside the abstract section, and "It appears in Computational Linguistics vol. ..." in the title section were removed.
- **Abstract boundaries:** Some files had *Introduction* or *Summary* instead of *Abstract*.
- **Authors, dates, university names**: All of these were deleted if they appeared inside the abstract or outside the BODY section.

5.3.6 From HTML to (simple) SGML

The HTML output file contains a lot of sophisticated tags which are not necessary for the subsequent processing of the texts. It also does not include end-of-paragraph markers, which are a requirement for SGML (every tag must end). At the end of processing, the only tags that should be present in the texts are[8]:

Root entity	`<SUMMARY-SGML>`
Document title	`<TITLE>`
Classification	`<CLASSIFICATION>`
Abstract section	`<ABSTRACT>`
BODY section	`<BODY>`
Headers titles	`<H1>`,`<H2>`, ...
Paragraph markers	`<P>`

and the hierarchical organisation of the text is:

[8] For headers, the higher the index the more general the header. This coding can store the structure of chapters, sections, subsections etc.

```
SUMMARY-SGML  →  | TITLE
                 | CLASSIFICATION
                 | ABSTRACT
                 | BODY          →  | P
                 |                  | H1,H2,...
```

The conversion is accomplished with several Perl scripts.

5.3.7 Post-Editing the SGML files

Even after the running of the previous script there were some files which needed extra editing because of:

- **Very strange and rare TAGS**: Usually LaTeX2HTML-produced tags that appeared infrequently and the Perl scripts did not take care of them in the first place.
- **Empty paragraphs**: Sometimes paragraphs without any text are produced by LaTeX2HTML. Besides their apparent usefulness, they tend to confuse the system in further stages[9].
- **Headers inside paragraphs**: Of course headers should not enclosed in paragraphs, they stand on their own.
- **Items lists**: Sometimes the items in a list do not end with full points, so the system considers them as a whole big sentence. We separated lists from the previous and next sentences as well as tagged every item as a sentence.

5.4 Tagging the corpus

The corpus so far is paragraph-marked, but we also need sentence, word and noun-phrase marking in order to count their frequencies. For assigning grammatical categories to the texts, I used the LTG tagger, a Hidden Markov Model based tagger with a correctness rate up to 96%. The inserted tags were `<S>, <W> and <NG>`, referring to sentences, words and noun groups (LTG term for noun phrase) respectively.

Here is a sentence before being tagged:

```
The formalism of synchronous tree-adjoining grammars
[[#REF]], a variant of standard tree-adjoining grammars
(TAG), was intended to allow the use of TAGs for language
transduction in addition to language specification.
```

[9] Of course empty paragraphs have a negative effect on heuristics based on location of sentences.

and after tagging:

```
<S ID="1"><NG> <W C="AT">The</W> <W C="NN">formalism</W></NG>
<W C="IN">of</W><NG> <W C="JJ">synchronous</W>
<W C="VBG">tree-adjoining</W> <W C="NNS">grammars</W></NG>
<D>[</D><D>[</D> <W C="CD">#REF</W><D>]</D><D>]</D><CM>,</CM>
<NG> <W C="AT">a</W> <W C="NN">variant</W></NG>
<W C="IN">of</W><NG> <W C="JJ">standard</W>
<W C="VBG">tree-adjoining</W> <W C="NNS">grammars</W></NG>
<D>(</D><NG> <W C="NN">TAG</W></NG> <D>)</D> <CM>,</CM>
<W C="BEDZ">was</W> <W C="VBN">intended</W> <W C="TO">to</W>
<W C="VB">allow</W><NG> <W C="AT">the</W>
<W C="NN">use</W></NG> <W C="IN">of</W>
<NG><W C="NNS">TAGs</W></NG> <W C="IN">for</W>
<NG><W C="NN">language</W> <W C="NN">transduction</W></NG>
<W C="IN">in</W> <NG><W C="NN">addition</W></NG>
<W C="IN">to</W>< NG><W C="NN">language</W>
<W C="NN">specification</W></NG> <EOS>.</EOS> </S>
```

Explanations

- The `ID` inside `S` is an attribute which refers to the index (position) of the sentence in the text.
- The characters in quotes inside `<W C=...>` refer to grammatical categories.
- The tags `CM` and `D` are used for non-word strings such as parenthesis, commas, etc.
- `EOS` denotes the end-of-sentence tag.

When this stage is finished, the corpus is ready for frequency counting.

5.5 Final statistics

Here are some indicative statistics concerning the corpus:

	TRAINING CORPUS	TEST CORPUS
Number of papers	204	5
Number of sentences	37K	736
Number of words (+titles+headers)	819K	15.3K
Number of words in body texts	810K	15.1K
Number of words after deleting functional words	371K	7.5K
Number of different words	23.6K	2.5K
Number of different words after deleting functional words	22.9K	1.8K

and the abstracts (for all the papers):

Average sentences per abstract	4.75
Average words per abstract	108.4
Average words per sentence in abstract	22.8

Chapter 6

Description of the system

6.1 Introduction

The basic schema under which the system tries to refine itself is that the same statistical model (a slight variation of TFIDF) will be tested on various representations of the text. We begin with single word representation and use TFIDF weighting to score the sentences and produce extracts.

In the next level, we attempt to improve the performance by adding more sophisticated representations (i.e. diagnostic units). First word pairs inside a sentence are used, a diagnostic unit which takes into account the context of words and therefore reduces the ambiguity problem present in the previous method. Second we take the notion of 'term' to the level of 'noun phrase' so as to minimise word ambiguity and capture terms beyond one single word.

Finally we try all the methods in all possible combinations. This produces 3 single methods, 3 paired methods (words + pairs, words + NPs, NPs + pairs) and 1 triple (words + pairs + NPs).

It is of high importance that the model is independent of the representations because it allows for more representations to be tried out without altering the model and generally the philosophy of the system.

The performance results will show (1) which combination produces the most satisfactory extract and (2) which representations the TDIDF statistical model can exploit.

6.1.1 Statistical model

The statistical model used to determine the diagnostic units weights is a TFIDF-like formula. Instead of the inverse document frequency, the system uses the inverse total frequency in a corpus (hence the name TFICF: Term-Frequency Inverse-Corpus-Frequency):

$$TFICF_i = \frac{\log\text{ (number of occurrences of term } t_i \text{ in the document)}}{\log\text{(number of occurrences of term } t_i \text{ in the training corpus)}}$$

The ratio of frequencies in a document over the whole corpus has the effect of supporting terms with high frequency in the document but rare in the rest of the corpus[1] while it is moderate with words that have the same frequency over both the document and the corpus[2]. The scoring function is implemented in such a way that no term gets a weight more than 1, even if the term frequency is bigger than the corpus frequency[3]. This adjustment was made to avoid extremely large scores for terms appearing only in the input text. Suppose for instance that a word appeared 5 times in a text and twice in the training corpus. Then it would have a weight of log(5)/log(2)=2.32, much above the normal weights. If this very high score should be kept is however a matter of preference; maybe such a term could be so valuable it deserves such a score. I personally favoured the normalised weights for two reasons: First, I found many similar cases not only because these terms were too specific and important for the text but also because of the small size of the corpus that allowed for many zero-frequency terms. Second, the problem was magnified for pairs and noun phrases since these diagnostic units had far less appearances.

6.1.2 Algorithm steps

There are basically two steps involved in the application of each one of the methods: (1) preparation of the corpus statistics and (2) the sentence scoring and sorting. The statistics (i.e. frequencies of diagnostic units) are collected from all the training documents beforehand. After this, the only statistics needed to be collected are the input text frequencies.

6.2 Method 1: Key words

6.2.1 Processing Steps

Collecting statistics over the whole corpus

The system needs to be trained by gathering statistics from the training documents. The procedure is as follows:

1. For all the documents in the training set compute the frequencies of all their words.

2. Remove all the words (functional+others) which do not contribute to the content of the sentence, by consulting a stoplist.

[1] This is a statistical way of stating that a word is a term for a particular document.

[2] Statistically speaking, such a word is an equally common word in all the documents, thus it does not have any discriminating power.

[3] Also, a term was scored with 1 if it there was no corpus frequency at all, or frequency of 1 because of log(1)=0.

3. Remove all the words with frequency 1, assuming that they are too specific or might be misspellings.

4. Sort the list by frequency in decreasing order.

Producing the extract

After the frequency table is constructed, the sentences of the test document are scored in the following way:

1. Calculate word frequencies for the words in the document.

2. For each sentence i compute its score s_i using the formula:

$$s_i = \frac{1}{N} \sum_{j=1}^{n} \frac{\log DF_{ij}}{\log CF_{ij}}$$

   where DF_{ij} is the frequency of the j-th word of the sentence over the document, CF_{ij} is the frequency of the j-th word of the sentence over the corpus and N is the number of content words in the sentence (i.e. its length minus the stoplist words).

3. Exclude sentences with less than 3 content words[4].

4. Sort the sentences by score in decreasing order.

5. Create the extract by selecting a certain proportion of the sorted sentences, e.g. 10% or only ones that are above a predefined number.

6. Display extracted sentences in the order of their appearance in the document.

6.3 Method 2: Word pairs

6.3.1 Special issues

Stoplist words

In the same way as the stoplist words for the first method, there can be certain pairs with one or even two insignificant words. These pairs are excluded from the further process, which also speeds up the system.

[4] In order to reject very short sentences which happened to have one or two very strong terms.

Frequency threshold for word pairs

For every sentence, of length n, there are $(n-1)!$ possible word pairs. It is an extremely large number which eventually results in a huge frequency table. In order to be able to store and access quickly the table, a decision was made to cut off pairs with very low frequency, assuming that their absence cannot affect the performance of the system[5].

6.3.2 Processing Steps

Again two steps, preparation and computation are required.

Collecting statistics over the whole corpus

1. Compute the frequency of potential word pairs, i.e. all the possible pairs of every sentence in the collection of documents. Pairs with words from the stoplist, or with word frequency less than 5 are excluded, the latter ensures the reliability of the statistics.

2. Remove all the words with frequency 1, assuming that they are too specific or that they are spelling errors from the corpus.

3. Sort the list by frequency in decreasing order.

Producing the extract

Basically the same formula is used to compute the sentence scores:

1. Calculate word pairs frequencies for the sentences in the document. If a word pair exists in both orders, use the total of both the two frequencies.[6].

2. For each sentence i compute its score s_i using the formula:

$$s_i = \frac{1}{N} \sum_{j=1}^{n} \frac{\log DF_{ij}}{\log CF_{ij}}$$

3. Exclude sentences with less than 2 word pairs[7].

4. Sort the sentences by score in decreasing order.

[5] See also section 3.1.1 for explaining why a threshold of 5 is needed for the reliability of the statistics.

[6] Naturally we assume that the order of words is not significant.

[7] Again a threshold (though a lower than the one for single words ensures that the extracted sentences are not very short.

5. Create the extract by selecting a certain proportion of the sorted sentences.

6. Display extracted sentences in the order of their appearance in the document.

6.4 Method 3: Noun phrases

The third method explores the possibilities of syntactic structures improving the system performance. Within the notion of looking for *terms* it is a logical sequence to go for noun phrases after words and pairs.

6.4.1 Configurational issues

To utilise syntactic information, it is obvious that the text needs to be parsed, so the output noun phrases rely on the quality of the parser and the noun phrase grammar specification. It is very useful to be able to configure the NP grammar because complicated structures can be simplified or rejected. For instance, if a prepositional phrase modifies the head noun phrase, the former can be deleted so the real diagnostic unit is only the main noun phrase. In addition, we could only select the last two or three words of a noun phrase as the diagnostic unit, thus eliminating adverbs and adjectives which are far away from the noun, etc.

Processing steps

The only difference in the algorithm for collecting statistics has to do with preparing the statistics:

- If a noun phrase had more than three words it was truncated to the last three.
- If a noun phrase is contained in the stoplist, then it is excluded.

The extracting algorithm is essentially the same. Noun phrases with frequency of 1 are not used in scoring and sentences with less than 2 noun phrases are also excluded.

6.5 Combined Heuristics

After the heuristics are tried separately, they are tried cumulatively in search for the optimum combination. The scoring formula is:

$$s_i = \sum_{j=1}^{K} \left(\sum_{t=1}^{N_i} a_j w_{t_j} \right)$$

where K is the number of heuristics combined, N_i is the number of diagnostic units of sentence i, a_j is the contributing weight of the j-th diagnostic unit[8] and w_{t_j} is the weight for the occurrence of the diagnostic unit in the i-th sentence.

[8]they were set to 1, thus giving equal weight to all the individual diagnostic units.

Chapter 7

Results and Evaluation

7.1 Evaluation design

Evaluation for the present project consisted of running the extraction program on the 5 test texts for all the methods (including all their combinations) and then subjectively judging the quality of all the output extracts.

All the extracts had 10 sentences, a practical value given the average of pages of the test papers (6 pages) and the average number of sentences in the abstracts of the training corpus (4.75 sentences) average.

This judgement was performed as follows:

- For every sentence there was a score on a five point scale according to the relevance of the sentence[1]:

SCORE	RELEVANCE
5	VERY HIGH
4	HIGH
3	CONSIDERABLE
2	MODERATE
1	POOR
0	NONE

Table 1: Sentence ratings

Generally, if the same sentence appeared in extracts produced by different methods, it got the same score. However, if it generated a better relevance within the context of the extract its score was increased.

[1] Based more on the sentence significance by itself and less on whether the sentence was in any way related to the abstract.

- The points of the sentences for every extract are added together; the sum represents the total quality.
- For every method the scores of the texts are also summed up to produce a quality measure for a specific method.

7.2 Results

The following tables show the total scores for papers and methods (W stands for words, P for pairs, N for noun phrases):

PAPER	**METHODS**							
	W	N	P	W+P	W+N	P+N	W+P+N	**TOTAL**
1	24	22	17	25	21	27	26	162
2	2	5	5	2	3	3	2	22
3	2	9	8	1	5	4	5	34
4	4	19	15	19	8	14	11	90
5	31	27	17	28	30	24	28	185
TOTAL	63	82	62	75	67	72	72	

Table 2: Subjective quality results for extracts produced for the 5 test papers.

7.3 Findings and Discussions

A first look on the scores shows a consistency of methods over papers, i.e. scores for a certain paper had about the same boundaries for all the methods. Total scores on the horizontal axis seem to belong to three categories. Papers 1 and 5 got very high scores, many of them five-point sentences (i.e. perfect for extracting). Paper 4 had a moderate scoring and papers 2 and 3 scored very badly. This was mainly due to the nature of the texts. Papers 2 and 3 had a lot of example sentences with a lot of proper names (one paper had in fact Japanese examples). These names, which did not appear at all in the training corpus, were considered high value terms and achieved impressive scores without the sentence having other content words. As a result we can say that texts with many example sentences, fractions of programming code, etc. were not effectively summarised by our frequency-count model.

7.3.1 Co-selection

Another observation made by looking at what sentences were extracted is the co-selection of certain sentences by different methods. The following table shows for every paper how many sentences were co-selected by more than one method (starting from 7, i.e. selected by all methods):

PAPER	CO-SELECTED SENTENCES BY METHODS						
	7	6	5	4	3	2	TOTAL
1	0	1	2	2	8	7	20
2	2	1	2	5	0	2	12
3	0	2	0	6	2	6	16
4	0	1	3	4	3	5	16
5	0	3	2	4	1	6	16
TOTAL	2	8	9	21	14	26	80

Table 3: Number of co-selected sentences by all methods.

This table demonstrates some kind of regularity. We expect the number of sentences co-selected to be lower in the first column and then increase gradually. This actually happens, except a gap in sentences selected by 3 methods. If we calculate the number of individual sentences appearing in the extracts by removing all the sentences that were repeated (i.e. selected by more than one method) we can get a measure of overlap between the methods. The total number of sentences in all the extracts is: 5 papers * 10 sentences * 7 methods = 350. 205 of them were repetitions, so 350-205=145[2] were distinct sentences. Of these 145, 80 are repeated more than once (80 is the sum of the last row of table 3) and the remaining 65 appeared exactly once. Hence, the overlap is 80/145=55,1% which shows that more than half of the sentences was selected at least from one other method.

Weighted combinations

In combinations, methods were assumed to be equally weighted (see chapter 6, Description of the system) but actually words were slightly less weighted because of the following fact: The highest scoring sentences (with TFIDF, not in my evaluation) with single-words did not usually reach the maximum score, i.e. 1. However, sentence scores for pairs and noun phrases achieved more frequently the maximum score. Actually, for the sentences extracted the average sentence score, computed with TFICF, for words, pairs and noun phrases was 0.66, 0.99, and 0.92 respectively. Therefore, when adding the scores of individual methods, the contribution of words was on average 0.66/0.99=66% the contribution of pairs and 0.66/0.92=71% the contribution of noun phrases. The reason why the average scores for pairs and NPs were that high comes from the scoring formula. Units for these methods had

[2] For example, we know that 2 sentences appeared 7 times, so 2*6=12 sentences were repetitions. This is how the number 205 came out.

much less occurrences in the corpus; it was easier for a sentence to have 2-3 units with a weight of unity, so the total sentence score was again 1 (also these methods had a smaller threshold of diagnostic units, see again chapter 6).

The fact that the word-based method did not contribute equally in compound heuristics is welcome (though not planned originally); since words have the poorest performance, it is desired to decrease their contribution.

What is the best method?

The scores over methods now show that the NP method had by far the best performance (82 points in total, 7 points more than the runner-up, pairs+words). Noun phrases also had consistently high scores in every paper, being the first scoring method in 3 papers (no.2, 3 & 4). Another observation which supports NP heuristics, is that 2 runner-up methods, ranked 3th and 4th, are combinations with NP as a factor.

On the other hand, words and pairs seem to perform similarly. However, I would say that pairs are in fact a better predictor because they outperform significantly the single-word method in 3 out of 5 papers; the 2 remaining papers which show a preference for words are in fact the two very high scoring ones. In these 2 papers (the "easier" for summarisation), high scores result in differences between methods. If we normalise the scores of every method by dividing by the total score for every paper we get:

PAPER	1	2	3	4	5	TOTAL
METHOD						
NP	22/162	5/22	9/34	19/90	27/185	0.984
P	17/162	5/22	8/34	15/90	17/185	0.826
P+NP	27/162	3/22	4/34	14/90	24/185	0.706
W+P+NP	26/162	2/22	5/34	11/90	28/185	0.672
W+P	21/162	3/22	5/34	8/90	30/185	0.664
W+NP	25/162	2/22	1/34	19/90	28/185	0.637
W	24/162	2/22	2/34	4/90	31/185	0.509

Table 4: Normalised performance of all 7 methods.

This viewpoint again proves the dominance of the NP method, but this time word pairs are clearly the second best method, while the single-word is the poorest one. Combinations are placed between pairs and words. If a combination includes the word method, it always achieves lower scores than the same combination without words.

7.3.2 Comparison with NetSumm extracts

A comparison of the program extracts with the extracts produced by the BT NetSumm program did not reveal any reliable results. The proportion of condensation for NetSumm was usually more than 15%, while it was less than 10% for the system extracts. The mean co-selection of sentences is only 16%, but this is not a comparable measure given that we do not know how NetSumm extracts are produced. However, NetSumm passages are more coherent and do not contain example sentences.

The following table shows how many sentences were co-selected (1) for the NP method and (2) for the best performing method regarding the specific paper (see Table 1. A star (*) indicates that for the specific paper, NP was also the best performing method):

	CO-SELECTED SENTENCES	
PAPER	NP method	best method
1	2	2
2	2	2*
3	0	0*
4	1	1*
5	3	1

Table 5: Co-selection of sentences between the program and BT's NetSumm.

7.3.3 Conclusion

The final conclusions drawn from the evaluation are:

- Noun phrases are the best predictors for summary-worthy sentences.
- More sophisticated representations than single words help to improve the system's performance.
- Combinations of methods do not produce better results, but still noun phrases are involved in the best of them. Finally, these synthesised methods are far more satisfactory than the single words diagnostic unit.
- Some papers are more suitable for extracting, some are not, depending on the nature of the text. Simple counting of term occurrences may extract sentences with too specific words, like proper names from examples, programming code, etc.
- 55,7% of the sentences were extracted by more than one method, and generally scores of different methods did not have a great diversity.

7.4 Future Extensions

The system can be extended in many ways, from heuristics to statistical models. Here are some suggestions:

1. **Use other statistical models**, incorporating inverse document frequency for example. Furthermore, we can discriminate between terms whose total global frequency is the same but some appear in small and others in big files.

2. **Experiment with weights of methods** by finding what weights increase (and maximise) the score of the important sentences. Supervised training procedures can be used for this.

3. **Explore other heuristics** including titles, headers and perhaps the most promising feature: position within a paragraph.

4. **Use structured noun phrases** in order to group NPs such as *information retrieval* and *retrieval of information*. This can be done by storing NPs in a tree structure: e.g. head of the NP + complements.

5. **Exclude example sentences, items in lists** assuming that their significance is negligible. This has also the effect of "clearing" the frequency tables, removing too specific units.

6. **Group lower-cased and upper-cased instances of the same word** to increase the occurrence statistics.

7. **Utilise abstracts and target extracts**: Much information can be learnt from abstracts, such as words, style, cue phrases, length of sentences, number of sentences, etc.

8. **Use different evaluation techniques**. We can judge the system extracts with human-prepared target extracts. Klaus Zechner (Zechner 1995) conducted such an experiment, and his results showed a high correlation (mean precision/recall: 47%/54%) between humans and machines. Text similarity and/or sentence similarity also can be incorporated to refine the evaluation.

9. **Develop techniques to deal with anaphora** which usually requires text understanding (Jones 1993), but it can refine substantially the extract.

10. **Try the system in a different corpus** to see if it achieves better results, and if certain features (e.g., example sentences, the statistical model) improve/harm its performance.

7.5 Summary

We have described and developed a system for automatic summarisation of texts based on sentence extraction. The system finds the most significant sentences of a text by computing a relevance-score for every sentence. This score is based on the statistical regularities of words, word pairs and noun phrases in (1) the input text and (2) a collection of training texts (800,000 words) all in the general domain of computational linguistics. Output extracts were produced for 5 test papers, for all the three methods and their possible combinations. Evaluation scores were subjectively assigned to the extracts in order to find the method which produces the most satisfactory extract. Results from the evaluation showed a clear preference for noun phrases, good performance for pairs and poor performance for words.

Chapter 8

User manual

The system was developed in C++ under the Unix operating system. There are two basic programs, create_dus to create the necessary representations and docsum to do the extracting.

8.1 create_dus

Syntax:

```
cat <sgml_file> | create_dus [-p] [-n]
```

where ¡sgml_file¿ is the input file, marked up at least for words and sentences. Option -p creates all the word pairs in a sentence and -n all the noun phrases. These tags are stored under the <REP> tag which is separate from <BODY>. All the entries have a <W> tag for uniformity, but there is an attribute T (for type) which has a value of "p" for pairs and "n" for noun phrases. Here is an example for the phrase *An automatic extraction program* with both pairs and noun phrases (word pairs consisting of at least one stoplist word are of course excluded):

```
<W T="n">automatic_extraction_program</W>
<W T="p"automatic_extraction></W>
<W T="p">automatic_program</W>
<W T="p">extraction_program</W>
```

Notes: There is one noun phrase here and three pairs. For the NP the article *an* was not included because maximally three words can be in one NP. The same article does not also appear in any pair because it belongs to the stoplist.

8.2 docsum

After the creation of the representation of the text, and given the frequency tables `docsum` produces the extracted sentences.

Syntax:

```
cat <sgml_file> | docsum [-m <method(s)>] [-p <condensation>]
[-q <query>] [-u <unit_query>] [-b <body_query>] [-h]
[-f <frequency_table>]
```

All the options have default values, especially set for the single word method. Here are further explanations about these options:

sgml_file:

It is the input file, marked up at least for words and sentences.

methods:

Indicates the diagnostic unit(s) to be used. `w` is for words, `p` is for pairs, and `n` is for noun phrases. For combinations the corresponding letters are simply concatenated, for example `wn` tries words + noun phrases (default: w).

condensation:

Corresponds to the length of the extract. If it is less than 1 then it indicates the proportion of the sentences that will be extracted (for example 0.1 means extract 10%). If the value is more than 1 then it indicates the fixed number of sentences (default: 0.1).

query:

This value represents the SGML tag where the diagnostic units are stored (default: `<BODY>`, for the single-word method).

unit_query:

Units are the SGML entities which are scored and extracted (default: sentences `<S>`).

body_query:

The SGML entity which contains text marked-up only for words and not other representations. It is the tag used to access the sentences for displaying them (default: `<BODY>`).

frequency_table

It provides the filenames for the look-up frequency tables. For simplification reasons it is assumed that files for different diagnostic units have the same root filename but different extensions. The default values are `total.frq`, `total.ng.frq` and tt total.wp.frq for words, NPs and pairs respectively. The filename given with this option corresponds to the single-word table. It is then split into root and extension and `.ng.frq` and `.wp.frq` are added in order to construct the remaining filenames.

8.2.1 Output

The output of the program is an SGML document with all the non-qualifying sentences removed. It still contains paragraph markers, headers, etc. and a perl script was created to make the output look more natural. See Appendix 1 for "clear" results.

Appendix 1: Sample extracts

In the following pages I present some of the extracts used in the evaluation. I chose extracts from 2 papers, one "easy" (high scores) and one "difficult". The methods selected for presentation were are all the individual diagnostic units (words, pairs and noun phrases) and the combination of the best units (pairs and noun phrases). The title and the abstract of each paper is also included.

The number preceding each sentence is the sentence id.

Paper 1: Default Handling in Incremental Generation
Karin Harbusch, Gen–ichiro Kikui, Anne Kilger

ABSTRACT

Natural language generation must work with insufficient input. Underspecifications can be caused by shortcomings of the component providing the input or by the preliminary state of incrementally given input. The paper aims to escape from such dead-end situations by making assumptions. We discuss global aspects of default handling. Two problem classes for defaults in the incremental syntactic generator VM-GEN are presented to substantiate our discussion.

METHOD: WORDS

(22) In the following sections, we argue for the appropriateness of processing-conforming default handling.

(25) For the opposite case where the default must be withdrawn we have to mark all defaults.

(39) In incremental generation, most authors agree on the necessity of using defaults (see, e.g., (De Smedt, 1990; Kitano, 1990; Ward, 1991)).

(70) A disadvantage is the difficulty of providing consistency between default-caused and input-licensed processing.

(71) Alternatively, the ongoing processing can deal with the default values in an ordinary manner (processing-conforming default handling).

(98) Then effects of the respective defaults are withdrawn introducing the input-provided values into the system.

(108) So, we end up with a processing-conforming default handler for generation realizing repair by non-monotonic changes.

(141) Evaluating the default body, the system creates a V-object OBJ.

(158) This paper proposes a processing-conforming default handler for generation realizing repair by non-monotonic changes. (163) We have verified the advantages of processing-conforming default handling by imple-

menting a default handler for VM-GEN.

METHOD: PAIRS

(27) This representation allows for easy checking the coincidence between a chosen default and input given later.

(34) Later on, default handling is included and exemplified for two general cases.

(40) Nevertheless, they do not in sufficient depth answer the question of how to guide the processes of default handling and repair within a generator.

(70) A disadvantage is the difficulty of providing consistency between default-caused and input-licensed processing.

(76) The knowledge source that is used for default handling should provide the most plausible actions for a default situation.

(79) A default description has the following form:.

(98) Then effects of the respective defaults are withdrawn introducing the input-provided values into the system.

(156) This object is able to synchronize global repair.

(158) This paper proposes a processing-conforming default handler for generation realizing repair by non-monotonic changes.

(163) We have verified the advantages of processing-conforming default handling by implementing a default handler for VM-GEN.

METHOD: NOUN PHRASES

(36) Furthermore, we discuss how default handling can be adapted to multilingual generation, as required by the speech-to-speech translation system VERBMOBIL (Block et al, 1992).

(39) In incremental generation, most authors agree on the necessity of using defaults (see, e.g., (De Smedt, 1990; Kitano, 1990; Ward, 1991)).

(53) Obviously there is a trade-off between fluency and reliability: maximal reliability requires secure decisions and therefore leads to output delay.

(76) The knowledge source that is used for default handling should provide the most plausible actions for a default situation.

(84) For incremental systems, we propose to express the body as a specification of input increments.

(105) Dependency links between the immediate result of a default body and results of the influenced decisions allow for this identification. (106) The disadvantage of non-monotonic changes is the complexity of computation, e.g., supported by a truth maintenance system.

(110) The syntactic generator VM-GEN is a further development of TAG-GEN (Kilger, 1994) within the framework of VERBMOBIL, aspeech-to-speech translation system.

(112) VM-GEN is able to consume input increments of varying size.

(119) The two reasons for missing information necessitate different methods of treatment which nevertheless both can uniformly be integrated into regular processing.

METHOD: PAIRS + NOUN PHRASES

(36) Furthermore, we discuss how default handling can be adapted to multilingual generation, as required by the speech-to-speech translation system VERBMOBIL (Block et al, 1992).

(53) Obviously there is a trade-off between fluency and reliability: maximal reliability requires secure decisions and therefore leads to output delay.

(59) Therefore, possible alternatives are either to wait for the next input or to trigger default handling.

(76) The knowledge source that is used for default handling should provide the most plausible actions for a default situation.

(83) The default body describes how to continue processing with defaults in an adequate way.

(84) For incremental systems, we propose to express the body as a specification of input increments.

(105) Dependency links between the immediate result of a default body and results of the influenced decisions allow for this identification.

(106) The disadvantage of non-monotonic changes is the complexity of computation, e.g., supported by a truth maintenance system.

(108) So, we end up with a processing-conforming default handler for generation realizing repair by non-monotonic changes.

(158) This paper proposes a processing-conforming default handler for generation realizing repair by non-monotonic changes.

Paper 2: A Uniform Treatment of Pragmatic Inferences in Simple and Complex Utterances and Sequences of Utterances
Daniel Marcu and Graeme Hirst

ABSTRACT

Drawing appropriate defeasible inferences has been proven to be one of the most pervasive puzzles of natural language processing and a recurrent problem in pragmatics. This paper provides a theoretical framework, called stratified logic, that can accommodate defeasible pragmatic inferences. The framework yields an algorithm that computes the conversational, conventional, scalar, clausal, and normal state implicatures; and the presuppositions that are associated with utterances. The algorithm applies equally to simple and complex utterances and sequences of utterances.

METHOD: WORDS

(19) 3. Another way of allowing pragmatic inferences to be cancelled is to assign them the status of defeasible information.

(45) [#IMG] Figure: The lattice that underlies stratified logic.

(74) ([#IMG]) John does not regret that Mary came to the party.

(77) Figure: Stratified tableau for John does not regret that Mary came to the party.

(81) Figure: Model schema for John says that some of the boys went to the theatre.

(83) Figure: Model schema for John says that some of the boys went to the theatre.

(102) ([#IMG]) Either Chris is not a bachelor or he regrets that Mary came to the party.

(105) Chris is not a bachelor presupposes that Chris is a male adult; Chris regrets that Mary came to the party presupposes that Mary came to the party.

(108) Being a bachelor presupposes that Chris is a male, while being a spinster presupposes that Chris is a female.

(116) The models yield Mary came to the party; Chris is a male; and Chris is an adult as pragmatic inferences of utterance ([#CREF]).

METHOD: PAIRS

(42) The lattice shows a partial order that is defined over the different levels of truth.

(59) Pragmatic inferences are triggered by utterances.

(67) The model-ordering relation filters the optimistic interpretations.

(101) Consider the following utterances and some of their associated presuppositions ([#CREF]) (the symbol [#EQN] precedes an inference drawn on pragmatic grounds):.

(105) Chris is not a bachelor presupposes that Chris is a male adult; Chris regrets that Mary came to the party presupposes that Mary came to the party.

(106) There is no contradiction between these two presuppositions, so one would expect a conversant to infer both of them if she hears an utterance such as ([#CREF]).

(107) However, when one examines utterance ([#CREF]), one observes immediately that there is a contradiction between the presuppositions carried by the individual components.

(108) Being a bachelor presupposes that Chris is a male, while being a spinster presupposes that Chris is a female.

(109) Normally, we would expect a conversant to notice this contradiction and to drop each of these elementary presuppositions when she interprets ([#CREF]).

(133) ([#IMG]) [#EQN] John is a male adult.

METHOD: NOUN PHRASES

(4) To understand conversational implicatures associated with indirect replies one must consider discourse expectations, discourse plans, and discourse relations [[#REF], [#REF]].

(39) The lattice in figure [#CREF] underlies the semantics of stratified logic.

(91) ([#IMG]) Not many most all of the boys went to the theatre.

(94) The theory provides one optimistic model schema (figure [#CREF]) that reflects the expected pragmatic inferences, i.e., (Not most Not many Not all) of the boys went to the theatre.

(98) By adding the extra utterance to the initial theory ([#CREF]), uttered (went (all (boys), theatre)), one would obtain one optimistic model schema in which the conventional implicatures have been cancelled (see figure [#CREF]).

(126) At this point Jane realizes that Mary misunderstands her: all the time Jane was talking about John Pevler, the five-year-old boy.

(145) Green s algorithm makes use of discourse expectations, discourse plans, and discourse relations.

(152) As Green notices, in previous models of implicatures [[#REF], [#REF]], processing ([#CREF]) a will block the implicature generated by ([#CREF]) c.

(157) Answer ([#CREF]) a makes the necessary conditions for implicating no true, and the implication is computed.

(160) Therefore, the implicature at the end of the dialogue is that the conversant who answered went shopping.

METHOD: PAIRS + NOUN PHRASES

(39) The lattice in figure [#CREF] underlies the semantics of stratified logic.

(49) The satisfaction relation is split according to the three levels of truth into u-satisfaction, i-satisfaction, and d-satisfaction:.

(81) Figure: Model schema for John says that some of the boys went to the theatre.

(83) Figure: Model schema for John says that some of the boys went to the theatre.

(94) The theory provides one optimistic model schema (figure [#CREF]) that reflects the expected pragmatic inferences, i.e., (Not most Not many Not all) of the boys went to the theatre.

(98) By adding the extra utterance to the initial theory ([#CREF]), uttered (went (all (boys), theatre)), one would obtain one optimistic model schema in which the conventional implicatures have been cancelled (see figure [#CREF]).

(105) Chris is not a bachelor presupposes that Chris is a male adult; Chris regrets that Mary came to the party presupposes that Mary came to the party.

(108) Being a bachelor presupposes that Chris is a male, while being a spinster presupposes that Chris is a female.

(126) At this point Jane realizes that Mary misunderstands her: all the time Jane was talking about John Pevler, the five-year-old boy.

(157) Answer ([#CREF]) a makes the necessary conditions for implicating no true, and the implication is computed.

Appendix 2: Stoplist words

Here is a list of all the words that did not contribute to any scoring (as words, parts of pairs, or noun phrases). Only the lower-case words are shown, although the list contains the upper-case ones as well. The first 9 entries refer to the indicator tags which replace several LaTeX features (see chapter 5: Corpus):

#AMP	#CREF	#EQN	#GT	#HREF	#IMG
#LT	#REF	#TABLE			
a	about	above	across	after	afterwards
again	against	all	allows	almost	alone
along	already	also	although	always	am
among	amongst	an	and	another	any
anybody	anyhow	anyone	anything	anyway	anyways
anywhere	apart	appear	are	around	as
aside	at	away	awfully	b	be
became	because	become	becomes	becoming	been
before	beforehand	behind	being	beside	besides
between	beyond	both	but	by	c
came	can	cannot	cant	certainly	clearly
co	concerning	consider	could	course	currently
d	did	do	does	doesn	doing
done	down	downwards	during	e	e.g.
each	eg	eight	either	else	elsewhere
enough	especially	et	etc	even	ever
every	everybody	everyone	everything	everywhere	ex
except	f	far	few	fifth	figure
first	five	for	forth	four	from
further	furthermore	g	get	gets	given
gives	go	goes	going	gone	got
gotten	h	had	happens	hardly	has
have	having	he	hence	here	hereafter
hereby	herein	hereupon	herself	himself	his
hither	how	however	i	i'd	i'll
i'm	i've	i.e.	ie	if	ignored
in	inasmuch	inc	indeed	insofar	instead
into	inward	is	it	it's	its
itself	j	just	k	keep	kept

know	known	knows	l	last	least
less	lest	little	ltd	m	many
may	me	meanwhile	might	more	moreover
most	mostly	much	must	my	myself
n	near	necessary	neither	never	nevertheless
nine	no	nobody	none	noone	nor
not	nothing	now	nowhere	o	of
off	often	on	once	one	ones
only	onto	or	other	others	otherwise
ought	our	ours	ourselves	out	outside
over	p	per	please	plus	q
quite	r	rather	re	really	s
said	same	says	self	seven	shall
she	should	since	six	so	some
somebody	somehow	someone	something	sometime	sometimes
somewhat	somewhere	still	such	t	than
that	that's	thats	the	their	theirs
them	themselves	then	there	there's	thereafter
thereby	therefore	therein	theres	thereupon	these
they	third	this	those	though	three
through	throughout	thru	thus	to	together
too	toward	towards	twice	two	u
un	under	unless	until	unto	up
upon	us	used	v	very	via
viz	vs	w	was	way	we
well	went	were	what	whatever	when
whence	whenever	where	whereafter	whereas	whereby
wherein	whereupon	wherever	whether	which	while
whither	who	whoever	whole	whom	whose
why	will	with	within	without	would
wouldn't	x	y	yes	yet	you're
your	yours	yourself	yourselves	z	zero

Bibliography

Borko, H. & Bernier, C. L. (1975), *Abstracting concepts and methods*, New York : Academic Press.

Dunning, T. (1993), 'Accurate methods for the statistics of surprise and coincidence', *Association for Computational Linguistics*.

Edmundson, H. P. (1969), 'New methods in automatic extracting', *Journal of the Association for Computing Machinery* **16**(2), 264–285.

Finch, S. & Moens, M. (1995), Sista final techical report, Technical report, Human Communications Research Centre, Language Technology Group, University of Edinburgh.

Goldfard, C. (1990), *The SGML handbook*, Oxford University Press. New York.

Jones, K. S. (1993), 'What might be in a summary?', *Information Retrieval* pp. 9–26.

Jones, P. A. & Paice, C. D. (1993), 'A 'select and generate' approach to automatic abstracting'.

Kupiec, J., Pedersen, J. & Chen, F. (1995), A trainable document summarizer, SIGIR '95, pp. 68–73.

Luhn, H. P. (1959), 'The automatic creation of literature abstracts', *IBM J. Res. Develop.* **2**(2), 159–165.

McKeown, K. & Radev, D. R. (1995), 'Generating summaries of multiple news articles', *SIGIR '95* pp. 74–82.

Mikheev, A. & Finch, S. (1995), Towards a workbench for acquisition of domain knowledge from natural language, *in* 'Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics.'.

Morris, A. H., Kasper, G. M. & Adams, D. A. (1992), 'The effects and limitations of automated text condensing on reading comprehension performance', *Information Systems Research* **26**, 17–35.

Paice, C. D. (1990), 'Constructing literature abstracts by computer: Techniques and prospects', *Information Processing & Management* **26**(1), 171–186.

Rath, G. J., Resnick, A. & Savage, R. (1961), 'The formation of abstracts by the selection of sentences: Part 1: sentence selection by men and machines', *American Documentation* **12**(2), 139–141.

Robertson, S. E. & Jones, K. S. (1994), Simple, proven approaches to text retrieval.

Salton, G. (1989), *Automatic text processing : the transformation, analysis, and retrieval of information by computer*, Reading, Mass. Wokingham : Addison-Wesley.

Salton, G., Allan, J. & Buckley, C. (1993), Approaches to passage retrieval in full text information systems, Technical report, Department of Computer Science, Cornell University.

Strzalkowski, T. (1992), Information retrieval using robust natural language processing, *in* 'Speech and Natural Language. Proceedings of a workshop held at Harrimon, NY', pp. 206–211.

Thompson, H. S. & McKelvie, D. (1996), 'A software architecture for simple, efficient sgml applications. the lt nsl software library.', LTG Presentations at SGML-Europe'96. Language Technology Group, Human Communication Research Centre, Edinburgh University.

Zechner, K. (1995), Automatic text abstracting by selecting relevant passages, Master's thesis, Centre for Cognitive Science, University of Edinburgh.