An application for the automatic generation of presentation slides from text

Mark Brown

BSc Computing
2004/2005

Summary

The aim of this project is to produce a system to semi-automate the process of generating slideshow content. The creation of computerised slideshow presentations for conference talks, currently requires a presenter to review their article and to manually condense the content into bullet point form. This system automates the process by converting an article into a set of slides, each containing a heading and a list of associated key-terms. An export facility allows users to save the generated slide content in a popular presentation application (OpenOffice's Impress). The developed solution required an in-depth understanding and adoption of the NLP technique of Term Extraction as well as Information Retrieval. The system is currently trained on a corpus of scientific articles, though the use of domain independent methods means that minimal alterations are required to extend the system to function with articles in other domains. The metrics of precision and recall were used for determining the accuracy of the implemented term extraction method.

Acknowledgements

I would like to express my gratitude to my supervisor, Katja Markert, for all her help and support offered throughout this project. Her extensive knowledge in the field of NLP, the critical feedback given for all draft chapter submissions and her responsiveness to queries by email were invaluable to understanding and completing all project deadlines.

I would like to thank all individuals who responded to my forum postings and emails regarding the use of the NLTK toolkit.

A final thanks goes to my friends and family for providing constant motivation and assisting in proof reading the final version of this report.

Contents

Section 1: Introduction

1.1 Problem

The task of producing presentation slides from one or several written materials is both tedious and time-consuming. The information encapsulated in slides typically includes key terminologies or summaries taken from a number of written materials.

The approach of finding key terminologies in a given text presupposes that a presenter has a certain appreciation of the subject area and thus is able to identify key-terms that are particular to the source. Another approach taken is to summarise the source material by extracting the most important sentences. As sentences are inappropriate for the direct inclusion into slideshows, they are condensed into bullet points which best characterise the sentence(s). The content of a presentation usually covers several topics or subject matters, each segregated into a number of slides. Having extracted key-information from the source text, in order to present this content in slideshow format, some means of segregating the key-information into different topics is required.

Automating the task of creating a slideshow from a given source would save presenters valuable time allowing them to concentrate on other aspects of the presentation such as preparing their speech. Although the ideal solution would be able to cope with a multitude of subject-domains and produce presentations of varying length, due to the time scale given for completing this project, a number of constraints need to be outlined to keep the scope of the solution manageable. The solution will work specifically on articles in the scientific domain. The reason for choosing a scientific corpus is purely down to ease of availability and as the solution uses domain-independent techniques, corpora from other domains could easily be handled. The duration of the generated slides will be limited to talks no greater than 15-20 minutes in length. This is the normal length of a conference talk and producing content for longer durations would be more problematic for the task of topic/text segmentation, which as can be seen after reading section 2.3, is an essential stage in presenting the extracted key-information. A final constraint will be that only one source text will be permitted for producing a presentation. This constraint represents a considerable simplification and is necessary for feasibility reasons to develop a solution in the timescale allowed.

This project will extend the writer's knowledge of Artificial Intelligence, focussing heavily on the field of Natural Language Processing. Throughout the writer's Computing Degree, AI has been a core competency, having studied AI11 (Introduction to AI), AI22 (Fundamentals of AI) and AI31 (Computer Vision). In addition, the topic of Information Retrieval learnt through DB32 (Technologies for Knowledge Management) and the use of regular expressions acquired during SY23 (Internet Systems Technologies) will both prove useful for understanding and implementing the system.

1.2 Project Aim and Objectives

The aim of this project is to design, implement and evaluate a method for automatically generating slides from text, to assist presenters in conferences in producing a presentation.

The objectives are:

1. Background research into the field of NLP.

2. Collation of a corpus of scientific articles.

3. Implementation of a tool/system to generate a slideshow from a written source.

4. Testing and Evaluation of the system against both human judges, and the title and abstract of documents in the scientific articles corpus.

1.3 Minimum Requirements and Extensions

The minimum requirements for this project are:

1. Overview and evaluation of the applicability of current IR and terminology extraction techniques for the generation of slideshows.

2. Design and implementation of a key term (words, terms, and formula) extraction method for identifying the most relevant content for a talk.

3. Implementation of the user interface for the extraction method offering the flexibility of changing the quantity and length of key-terms to be returned.

4. Collection of a sample corpus of articles/papers.

5. Evaluation of system to sample corpus.

Possible extensions are:

1. Implementation of a method for sentence extraction (summarisation).

2. Displaying generated content in slide format.

3. Converting raw-texts into marked-up XML documents.

1.4 Deliverables

The two deliverables for this project are:

- The Project Report
- System for generating a slideshow from a written source

1.5 Project Management

The project plan outlines deadlines marking the end of significant stages of this project. As the distribution of modules studied during the writer's final year is unequally balanced (5 in the first semester and 3 in the second), the structure and time-scales allow for this by weighting the workload towards the second semester. The key-stages of the project plan and their projected end dates are:

- Background Reading (29/11/2004)
- Complete Design section of Report (10/01/2005)
- Implement the key-word/term extraction method (21/02/2005)
- Implement Front-end and any additional extensions (14/03/2005)
- Test system against Title and Abstract (21/03/2005)
- Test system using Human Judges (28/03/2005)
- Complete evaluation (18/04/2005)
- Complete and submit final report (27/04/2005)

At the point just after completing the initial project plan, a significant alteration was made concerning the number of iterations of the life cycle to be undertaken. Having researched the main system processes and identifying that the complexity of implementing the core functionality would only allow for one iteration (using SDLC), a revised project schedule was produced. The above deadlines and the project schedule in Appendix B, reflect the revised schedule.

Section 2: Background Reading

To implement a system able to generate slideshow content, a certain depth of understanding is required in the NLP fields of corpora, key-word extraction and topic/text segmentation. To avoid any ambiguity that may occur in interpreting some of the fundamental terminology, precise definitions need to be given as a prerequisite to understanding the content of this report. When talking about the source text/material/document, this refers to the text on which a presentation will be based. Separately to the source document, the terminology 'talk' in the context of automatically generating slides, refers to the presentation slides that have been used in real talks. When testing the system, unseen texts taken from the scientific article's corpus will be used. By unseen, the writer means documents belonging to the corpus, which are not used as part of the training set.

In order to extract information that best identifies a document, the solution must build a knowledge base of the key-terms associated with a specific domain. A corpus (collection) of documents sharing the same domain as that of the document being used for the presentation is required. A list of domain specific terms is then constructed from the corpus and utilised in comparing with the terms identified in the source material. This allows the system to determine those terms that best embody the semantics of the source document.

Identifying which terms are to be extracted from both the corpus and presentation source requires an in-depth look and appraisal to be made for the various NLP key-term extraction methods. A discussion into the applicability of each method will then form the basis for concluding which method(s) will be adopted for the task of key-term extraction.

Providing the option of viewing key-sentences as supposed to just words and phrases allow presenters to make their own judgements of what text will be used to formulate the presentation. Although this approach means presenters will have to spend a greater amount of time converting the sentences into slide-friendly content as opposed to if they had just been given the terms, the scope for producing the content will be enlarged. Due to the time scale of this project, only key-terms (not key sentences) will be identified from the source document.

For the system to be able to generate slides based upon the information extracted from the source, some method by which the text can be analysed and segregated into topics needs to be adopted. This requires researching and utilising a method for topic/text segmentation, which will allow identified key-terms to be split into a number of topics and thus slides.

2.1 Corpora

2.1.1 Overview

Corpus is the Latin word literally meaning ‘body’. In the context of computational linguistics, a corpus is a collection (body) of documents that are usually in machine-readable format. McEnery and Wilson (2001) define four broad categories/components for defining corpora within modern linguistics as ‘sampling and representativeness’, ‘finite size’, ‘machine-readable form’ and ‘a standard reference’.

There are two approaches to sampling and compiling corpora. If the field being analysed has few texts, all texts in the field should be collected as the basis for forming the corpus. If on the other hand, a domain is chosen having a larger total text population, a sample should be chosen in a way that will maximally represent the subject matter (domain) it is attempting to characterise.

When constructing the corpus, a finite restriction should be placed defining the number of documents to be collected and the maximum total word count in each case. A predefined maximum word count for the entire corpus can then be calculated and upon reaching this threshold, the collection process stops and the collated samples form the corpus.

Although originally, corpora were mostly presented in printed text format, nowadays the term usually implicitly refers to a collection of documents in machine-readable format. Storing corpora in this form allows software to be implemented to manipulate the text in ways, which would take a considerable while longer if undertaken manually by people.

The final component is that corpora should be a standard reference for the domain it is characterising. This implies that the corpus be widely available for use by the linguistics community, providing a benchmark for future studies.

In the field of NLP, corpora have been used to improve spelling correctors, hyphenation routines and grammar checkers. Corpora used in NLP, usually all relate to a specific domain (may be multi-lingual) or characterise a language (using several domains). Corpora can come in the form of raw text i.e. plain text with no additional information or as is the case in the corpus collection for this project, a mark-up scheme can be used to provide additional linguistic information about the text. The most common mark-up schemes are SGML and XML.

2.1.2 Corpus of Scientific Articles

Although the solution will adopt generic domain-independent methods to solve the task of generating a slideshow from a text source, a set of training data specific to a genre is required to assist in finding information most relevant to the genre of the source document.

The corpus to be used in this project is a collection of scientific articles collated by Simone Teufel. Originally, the corpus was used for developing *"a new type of analysis for scientific texts"* (Simone Teufel, 1999), called 'Argumentative Zoning'. Being a readily available, large and a marked-up (XML) corpus, depicting a specific domain, makes it appropriate for reuse in this project. The scope of source documents will as such be limited to the genre of science. The corpus has been marked-up using a DTD (Document Type Definition) scheme and XML tags to depict the structure of the document. The tags add general details about a document such as its title, authors and date of publication, but also add formatting information. This includes identifying sub-headings, start and end of paragraphs/sentences and references. In total, there are 200 documents in this corpus of which 80 will be used to comprise the CorpusSciTR (training set). Each article contains around 3,000 - 4,000 words of an average word size of 6 characters. The additional information provided by the mark-up scheme will be a valuable aid in segregating the content extracted from the source into smaller sized units to be placed on different slides.

2.2 Key-word Extraction

There are three distinct methods in use within the field of NLP for identifying key-words in a passage of text. The essence of both collocations and term-extraction techniques is to identify words and phrases in the source text and/or document collection. Terminology discovered can then be used by IR systems to find the most relevant documents/terms belonging to the source document (more usually known as the query expression).

An important concept used in all three methods is the notion of terminology, which is a sequence of words whose meaning contributes to the characterisation of a specific domain or document. Extracting terminology through identifying collocations requires searching a document for units of two or more words and through statistical analysis, determining those phrases whose words are found to co-occur with a probability greater than mere chance. Term extraction, discovers terminology by identifying phrases whose sequence of part of speech (POS) patterns is identified against a pre-defined POS sequence(s) as having the structural properties that constitute terminology. To determine the most appropriate technique(s) for generating the content of slideshows, a detailed review of the three methodologies will now be discussed.

2.2.1 Collocations

2.2.1.1 Overview

Extracting key terminology through identifying collocations requires a genre specific knowledge base to be formulated from the CorpusSciTR for use in extracting collocations in the presentation source. Collocations differ from the identification of terms through term extraction in that they consist of two or more words (term extraction includes single word units). The rest of this section will discuss how collocations are identified and extracted from a document and the applicability of the method in the context of extracting key-words/terms for the basis of slide content.

A collocation as defined by Manning and Schutze (2000, pg. 151) is any *"expression consisting of two or more words that correspond to some conventional way of saying things"*. Identified expressions include noun phrases ('strong tea', 'pitch black'), phrasal verbs ('to wash up', 'to push in') and other stock phrases ('the rich and famous'). They propose that any combination of words, which cannot be directly translated word-by-word into another language and still hold exactly the same semantics, is a strong candidate for a collocation. Frank Smadja (1993) adds that collocations are *"recurrent combinations of words that co-occur more often than expected by chance and that correspond to arbitrary word usages"*. This implies that collocations are identified through the frequencies of all words belonging to the term occurring greater than by mere chance. An example is that supposing the term 'broad daylight' was identified as a collocation, although the chance of each word occurring on its own is quite likely and thus the chance of the words co-occurring by chance would also be relatively high, by comparing the expected number of co-occurrences of 'broad daylight' (occurring by mere chance) with the actual number of co-occurrences in the text, a conclusion can be drawn that a particular phrase does indeed form a collocation.

2.2.1.2 Methods for extracting Collocations

2.2.1.2.1 Frequency

The frequency method for extracting collocations, involves counting the number of occurrences of n-grams. This method alone is insufficient, as many terms would be returned that are compositional. The solution proposed is to filter out phrases using what is known as a 'part-of-speech' filter, which only accepts terms, whose *"patterns are likely to be phrases"* (Manning and Schutze, 2000, pg. 155). Although this method is simple and intuitive to implement and when combined with the part-of-speech filter, produces very accurate results, its limitation lies in only being able to identify adjacent words as terms.

2.2.1.2.2 Mean and Variance

This method takes into account the fact that collocations may occur where the individual parts (words) do not necessarily lie adjacent to one another, but rather at varying distances. The

phrase 'knock … door' for example contains a relationship whose words occur at varying distances from each other depending on the context of the phrase (*'I* knocked *on the* door*'* or *'A man* knocked *on the newly installed front* door*'*). The first principle is to set up a collocation window, which contains a certain number of words, each side of a word. Then dependent upon the size of the collocation being formed, all possible combinations of words for the chosen size are constructed from the window. This larger list of n-grams is then used by taking each n-gram and computing the offsets between the occurrences of all words. The mean is simply obtained by the average offset and the variance is a measure of the amount by which the offsets vary from the mean. Those n-grams having a low deviation can be said to form a collocation around the mean offset. Smadja (1993) comments that the mean and variance method for extracting collocations has an accuracy of about 80%. The operation of creating a larger set of n-grams through the use of a collocation window will however, take a greater amount of processing time and thus increase latency.

2.2.1.2.3 Hypothesis Testing

As Manning and Schutze (2000) comment, the methods discussed thus far don't consider that a *"high frequency and low variance can be accidental"*. To determine whether all words in an n-gram occur more significantly than by chance, a null hypothesis (H_0) needs to be devised that there is no association between the words of an n-gram other than by mere chance (independence). The calculated H_0 is then compared with the actual number of occurrences of an n-gram and by using significance testing, a pre-defined set of critical values determine whether the n-gram should be regarded as a collocation. In the context of this project, hypothesis testing allows collocations to be ranked and would offer the user extra flexibility and control in being able to restrict the number of key terms to be returned from a text.

2.2.1.3 Evaluation

In order to measure the quality/accuracy of the collocations returned by the system for characterising the source document, criteria is required that should hold for all proposed collocations. Manning and Schutze (2000, pg. 184) further qualify what exactly constitutes a collocation in the form of the following criteria:

- **Non-Compositionality** – *"The meaning of a collocation is not a straight-forward composition of the meaning of its parts"*.
- **Non-substitutability** – *"We cannot substitute other words for the components of a collocation even if in context, they have the same meaning"*.
- **Non-modifiability** – *"Many collocations cannot be freely modified with additional lexical material or trough grammatical transformations"*.

To evaluate the accuracy of a system identifying terms using collocations, a comparison can be made with the result of manually extracting all possible n-grams from the source document and applying the above criteria. Due to the complexity of utilising the defined criteria, one or more

human evaluators will be required since deploying a computer program able to compare the semantics of words is not the focus of this project.

To derive the set of collocations meeting the above criteria from all possible n-grams in the source document, human evaluators should first apply the non-compositional criterion. Through their own experience, they would identify those n-grams whose combined meaning of the words cannot be derived by the sum total of their individual meaning e.g. 'stark reality'.

The next step is to take the compositional phrases and then identify those, which are non-substitutable phrases. Human evaluators would again review the list of phrases and in each case through their own experience; identify those phrases whose words cannot be substituted with other words and still hold the same meaning. Where no other semantically similar combination can be obtained, a non-substitutable phrase has been found.

Due to the time scale that would be involved in asking several human evaluators to manually apply the criteria, the final criterion of non-modifiability will not be considered in the context of this project. The resulting list from performing the above criteria forms the ideal set of collocations characterising a particular document. A simple measure such as precision can be used to calculate the percentage of collocations in the ideal set that also appear in the set of collocations generated by the system.

2.2.1.4 Conclusions

Collocation identification can be utilised to solve the task of extracting key-terminology from the presentation source by identifying collocations in the CorpusSciTR and separately identifying collocations in the source document. One statistical comparison would be to use IR to identify and rank for all collocations in the CorpusSciTR, those mostly occurring in the source document. As described in the evaluation for one possible IR solution (section 2.2.3.5), this approach has the drawback of failing to find those collocations that only appear in the source document but not in the CorpusSciTR.

2.2.2 Term Extraction

2.2.2.1 Overview

The field of term extraction takes quite a different approach to key-term extraction than the frequency-based methods used in collocations. A Term as described by Beatrice Daille (1994) is *"mainly multi-word units of nominal type that could be characterized by a range of morphological, syntactic or semantic properties"*. To extract terms, a linguistic approach is taken analysing the structural properties of text to identify terms against a pre-defined list of patterns. These patterns are usually part of speech patterns e.g. 'noun – preposition – noun', though other patterns such as string matches are also widely used. Another

notion, proposed by Sophia Ananiadou (1994) is that terms are usually domain specific, *"a term typically refers to a specific concept in a particular subject field"*.

2.2.2.2 POS Tagging

The goal of POS tagging is to correctly assign each word in a text with a part-of-speech label. As words can have more than one possible label, the context within which a word lies, needs to be analysed to make accurate decisions. The fundamental aspect of POS tagging is the use of a tagger. Taggers take corpora, which have already been tagged. The tagged corpora serves as the knowledge base (vocabulary), allowing taggers to proceed in tagging untagged documents/corpora. A popular though simplistic and error prone tagger is the Unigram Tagger. This assigns tags to words in an untagged document based upon the most frequently occurring POS tag for each given word in the CorpusSciTR. This principle alone however is insufficient, as a sizable portion of words would be inaccurately tagged. Manning and Schutze (2000) believe the inaccuracy stems from the fact that many content words in English can belong to multiple parts of speech. Other taggers such as the Brill Tagger take the document/corpora tagged by the Unigram Tagger and proceed to minimise tagging errors by finding rules by which to modify tags e.g. change all nouns to verbs if the preceding word is tagged 'TO' i.e. the word 'to'.

An alternative method for POS tagging is to use the Markov Model. A manually tagged training set is used to determine the probabilities of transitions between tags. These probabilities are then utilised by assigning tags to each word in the source document. For each word, the highest probability of a POS tag occurring given the previous tag is calculated. Having constructed the Markov Model, a method is required to determine the precise nature by which the presentation source will be tagged. The Viterbi algorithm tags a document and follows the steps of initialisation, induction and termination. At the very beginning of the text and for each identified 'full-stop', a period tag with the probability of 1.0 is assigned (initialisation). For each word occurring between periods, a maximum likelihood algorithm is applied to determine the tag to be assigned based upon the previous state/tag (induction). This is a somewhat simplified view as the actual implementation calculates the maximum likelihood for all words in a sentence before assigning any tags. This supposes that for a given word, where the POS tag assigned to the previous word is found to be highly unlikely, then the probabilities will change and this may cause alterations in the assignment of tags as the induction process proceeds. Termination occurs at the end of the induction process when a specific tag appears.

2.2.2.3 Methods for Term Acquisition/Extraction

As discussed in the previous section, POS tagging provides an effective solution for identifying key-terms from a passage of text. In general, term extraction systems identify key-terms by matching whole strings, lemmatised strings or POS patterns.

One popular system for term acquisition named TERMINO (Jacquemin, 1994), extracts terms through identifying noun-phrases. A domain-specific corpus is used to discover noun-phrases that are likely to appear in the target text. LEXTER (Jacquemin, 1994), another widely used term acquisition system relies upon a manually POS tagged corpus. The main principle is to identify noun-phrase boundaries against a set of pre-defined POS patterns. Each phrase occurring between identified boundaries forms the list of candidate terms. LEXTER filters this list of terms by the use of a module to compare and evaluate the ambiguous and unambiguous constructions.

Having identified a list of domain-specific terms from corpora, term extraction on a document sharing the same domain as the corpus can then proceed. The methods described above for acquiring terms from corpora can also be utilised for extracting key-terms from the source document. Each identified term is then ranked using a statistical method (weighting scheme in IR) as is also the case for the collocation extraction method to determine those terms that are most distinctive to the document.

2.2.2.4 Conclusions

The term extraction method is another effective method for extracting key-terminology in a given document. Unlike the collocation approach, which quantifies the probability of phrases occurring more than by mere chance, term extraction takes a more qualitative approach, looking for pre-defined strings and/or POS patterns that when used in conjunction with a stop list or other such filters/modules, discovers terms that are believed to be of significance to the document. Term extraction can be used with the IR solution as an alternative to collocations for identifying a list of phrases to be indexed.

2.2.3 Information retrieval

2.2.3.1 Overview

Information Retrieval is used by search engines to retrieve textual information. IR involves developing *"algorithms and models for retrieving information from document repositories"* (Manning and Schutze, 2000, pg. 529). Although in the context of this project IR refers to the retrieval of textual information, the techniques could also be applied to speech and images/video. The most frequently occurring IR operation is the ad-hoc retrieval. This involves returning documents, which match a query, commonly entered by the user. Although a simple implementation could just return documents matching exactly some query (Boolean Search), as Manning and Schutze (2000, pg. 530) point out, *"for large and heterogeneous document collections, the result set of exact match systems usually are either empty or huge and unwieldy"*. The rest of this section will discuss the process of taking terminology discovered from the document collection (CorpusSciTR) through techniques like collocations or term extraction and performing a match against a query expression to determine which documents (or phrases from the source document) should be returned. The section will end with a conclusion on how to adapt this technique to solve the problem of finding domain specific terminology from the presentation source.

2.2.3.2 Inverted Index

An Inverted Index is a table showing for each identified key-term, which documents contain that term and the number of occurrences (frequency) in each document. In addition to term frequency, many Inverted Indexes store positional information. This is calculated as the offset of a term in relation to the beginning of the document. Storing positions allows more sophisticated optimisation to take place when for example ranking documents taking into account the spread of all matched key terms in the document.

Manning and Schutze (2000) describe the use of a separate phrase identification module, which indexes documents based upon finding phrases, which are present in both the document and a pre-defined list of domain specific phrases. They propose that to compile the list used by the phrase identification module, candidate phrases can be elicited by using the technique of discovering collocations (see section 2.2.1) to find the most frequently occurring bigrams. A stop list is required to ensure function words ('is', 'like' and 'not' for example) are not indexed. Stemming words to increase the probability of matching individual words is another common feature of IR systems.

2.2.3.3 Vector Space Model

Ranking documents requires some means by which each document can be compared with the query. The vector space model is a popular, widespread method for ranking documents based upon a comparison of the frequencies of each word/term occurring in a document with the query. The principle behind the model is to construct an nth-dimensional space where each word/term in the document collection (CorpusSciTR) and the query (terminology elicited from the source document) corresponds to a dimension in this space. Vectors are then drawn, each representing the relative frequencies of all words/terms for a given document in the collection. A vector is also plotted for the query expression to allow comparisons with all other vectors to take place. The proceeding section (2.2.3.4), discusses how to effectively weight all the words/terms in each document as well as for the query expression, to plot the vectors.

Documents are then ranked through looking at the similarity of all documents (vectors) with the vector representing the query. The cosine measure ranks documents in descending order, starting with the document whose vector has the smallest angle with that of the query vector. The following calculation is used to determine the angle (cosine) between a document and the query:

$$\cos(q, d) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} d_i^2}}$$

q = query expression vector d = document vector i = the ith term of the document/query vector

The cosine is determined by the sum of computing for each term i, its similarity (difference in frequency) between the documents and the query, dividing the result by the Euclidean length of the two vectors.

2.2.3.4 Term Weighting

To understand how to go about weighting terms, the concepts of term frequency, document frequency and collection frequency need to be clearly defined. Manning and Schutze (2000) describe term frequency as *"how salient a word is within a given document"*. The TF shows the number of occurrences for a given term in a given document calculated by taking 1 plus the log of the number of appearances of a term in a document (1 + log(tf)). Document frequency is defined as *"as an indicator of informativeness"* (Manning and Schutze, 2000). This shows the number of documents in which a given term occurs out of the document collection. Document frequency like term frequency can also be expressed logarithmically. The most popular method used to express document frequency is the Inverse Document Frequency (idf).

$$idf = \log \frac{N}{df_I}$$

The calculation divides the total number of documents in the collection (N) by the document frequency for a given term (i) and takes the log of the result. The purpose of the log function for calculating both tf and idf is to ensure the importance (weighting) given to a term increases by smaller increments for every additional occurrence of a term. For example, if a term occurs twice in one document and four times in another, the document should not be considered as much as twice as important, but simply as just more important. The final concept, collection frequency is just a count of the number of occurrences of a term in the document collection.

Information Retrieval Systems calculate term weights using a scheme based upon TF.IDF. The term frequency (tf) and the inverse document frequency (idf) are combined to give an overall term weighting. Manning and Schutze (2000 pp. 543) say that all schemes based upon the TF.IDF can be characterised by *"its term occurrence weighting, its document frequency weighting and its normalization"*.

2.2.3.5 Evaluation

As there are many IR systems differing in implementation, researchers had to devise some mechanism for evaluating the quality of each system. Two measures, which are used extensively, are precision and recall. Manning and Schutze (2000, pg.534) define precision as *"the percentage of relevant items in the returned set"* and recall as *"the percentage of all relevant documents in the collection that is in the returned set"*.

One method to assess the precision of different IR systems is to calculate the precision for each document in the returned set. The calculation measures precision for each relevant document, but in each case, only considers those documents ranked in higher positions. A final score of the IR system is obtained by aggregating all the precisions. This method is known as the uninterpolated average precision. An alternative method, interpolated average precision calculates precision at 10% intervals of recall for each ranking usually from 0-100%. "*Precision is determined by the point in the ranked list where the proportion of retrieved relevant documents achieves a certain recall level"* (Manning and Schutze, 2000). Should precision increase whilst moving down the ranked list, the highest level of precision is taken as the resulting interpolated average.

2.2.3.6 Conclusions

Information retrieval is essential for returning documents relevant to a query usually entered by a user. In order to adopt IR for the task of extracting key-terms out of the presentation source, a hybrid solution must be used so that instead of returning documents of most relevance, a list of key-terms most relevant to the source document is returned. The solution would be to create an inverted index from words/phrases discovered in the CorpusSciTR, storing the total number of occurrences of each word/phrase for each document in which they occur. Having created the index, the next step proposed would be to formulate a query expression from the source document. On top of the normal stop and stemming processes, each word in the document may be looked-up in a thesaurus to find synonyms and this enlarged list of words and terms then formulate the query expression. The TF.IDF matching algorithm would then be adopted to determine for all phrases in the training set, those having the highest occurrence in the source text than compared with the CorpusSciTR. The drawback of this solution is that those phrases unique to the source document would never be returned.

2.4 Topic/Text Segmentation

2.4.1 Overview

The key information generated by the system, will at this point be unstructured and simply returned to the user as one long list. The aim of topic/text segmentation is to take a document and find logical points by which to segregate the text into chunks, each corresponding to a particular topic/subject-field. In the context of producing a system for generating slides, the source document would be scanned to find shifts in topics. Each returned unit of key-information would then be categorised based upon the place in the source text from which it was extracted. The system will then have a basic presentational ability to split the key information into a collection of slides. The rest of this section will discuss the implementation of TextTiling for identifying shifts in topics in the source document.

2.4.2 TextTiling

The TextTiling algorithm identifies parts of text in a given document where vocabulary usage shifts from one topic to another. Each shift in a topic corresponds to a boundary of a multi-paragraph unit focussing on a specific (sub-) topic. The algorithm defines four stages by which to identify these multi-paragraph units.

The first stage is to split the document into what is known as *token sequences*. These token sequences can be of any arbitrary length, but an optimal length as proposed by Hearst (1997) is somewhere around 20 words. The points between tokens are referred to as gaps.

A *cohesion scorer* then computes each gap's 'topic continuity' i.e. measure the amount by which the same topic is resident in tokens lying both sides of a gap. A low cohesion score signifies a possible shift in topic. One means by which topic continuity can be calculated is to take a number of tokens each side of a gap (Hearst, 1997 suggests 2 tokens each side) and compute the correlation coefficient of their term vectors through using weighting schemes as discussed in the term weighting section (2.2.3.4) for IR.

Having computed cohesion scores at each gap, the next step is to calculate a *depth score* for each gap. This involves assigning a depth score based upon comparing the cohesion score of a gap with that of its surrounding gaps. The idea is to find those gaps having a lower cohesion score than with that of its surrounding neighbours (resulting in being assigned a high depth score).

The final stage, *boundary selection* is concerned with selecting those gaps with the highest depth scores as points with which to segregate text in a document. This is done by finding both the mean (μ) and standard deviation (σ) of all depth scores. Boundaries signifying changes in topics, are identified by finding those gaps having a depth score greater than:

$\mu - c\sigma$ (where c = constant for weighting S.D.)

Section 3: Methodologies

Now that the topics and methods relevant to this project have been identified, the tools, methodologies and the design of the system will be discussed during this chapter. The 'Methodologies' section is split into four parts, 'Programming Languages', 'Software Methodologies', 'NLP Methods' and 'User Interface Design'.

The programming language section looks at the benefits and drawbacks of the various applicable programming languages. A criteria is identified for what functionality/features the target language(s) need to support for the purposes of this project. Based upon the criteria, a decision is made as to the exact languages to be adopted.

Following on from the choice of programming languages, the 'Software Methodology' section reviews the choices of three possible software development life cycles. A justification is then provided as to which methodology will be utilised. Having selected a methodology, the alterations required to tailor the life cycle for the needs of this project is then discussed.

'NLP Methods' provides a justification as to the specific methods as discussed in the previous section (Background Reading, 2), which will be adopted in the implementation of the system.

Once the main functionality, which the system will provide has been identified, the final section (User Interface Design) will detail a framework (guidelines) by which all components of the UI will comply.

3.1 Programming Languages

3.1.1 Criteria for choice of Programming Languages

The task of choosing one or more programming languages for the implementation phase, required determining what features of programming languages were of particular importance to this type of system. The following four requirements define the desired features with which languages should support for this type of project:

1. Scalability – as the program will be processing corpora containing millions of words, the scalability and efficiency of the target language is of great importance.

2. Support for Regular Expressions – in identifying the main system processes, a common task that must be supported in the target language is the ability to perform pattern matching through the use of regular expressions.

3. Rapid Development – being a project having a relatively small time-scale and limited resources, due to the complexities of the implementation, the target language must allow for fast, timely development. Additionally, the writer has a preference for languages having NLP Toolkits to reduce the workload.

4. Previous Experience – To minimise the learning curve required during the implementation stage, the writer has a preference for adopting a language with which he is already familiar.

3.1.2 Evaluation of Programming Languages

In determining the criteria by which one or more languages may be chosen, an evaluation of several languages against the criteria is now given.

Python is a scripting language, which was developed in the early 1990s originally for use in academic institutions to teach new students, the main programming concepts. Being a high level language, Python allows for rapid development. This is desirable for implementing the complex core logic of the program. One of the many modules that comes as standard with Python is support for regular expressions. This is again of great importance for this project. In terms of Python's scalability, although not as quite as efficient as 'C', the language still offers faster processing speeds than 'Java'. A final mention should be given that there is a NLP library available for use with Python (NLTK). On further investigation, the NLP library offers many of the features that will be required according to the list of identified system processes and so will be of great value in reducing the workload.

Java is an object-oriented language, developed in the early 1990s, having a large community of users. Originally developed for use as a web technology (Java applets), Java has proven to be very popular also for application development. In having a large community, there is a vast collection of libraries to help make the implementation for programmers that little bit easier. One such package, which comes as standard with Java is a GUI framework. Having previous experience in developing GUI applications in Java, the writer is aware of the flexibility offered by the package. The front-end of systems can rapidly be developed in Java and can be customised to whatever level of sophistication the developer requires. Java is not very scalable in processing large quantities of data in one go. The emphasis is placed on performing large volume (small) transaction processing. This language is therefore not very appropriate for implementing the core logic of the program.

"Jython is an implementation of the high-level, dynamic, object-oriented language Python, seamlessly integrated with the Java platform" (*What is Jython?*, 2003). It allows full integration with Java e.g. seamless integration with Java's GUI classes. Although adopting Jython for implementing the core logic of the program and developing the GUI in Java would allow full interaction between the two, the main drawback is that Jython is simply not as efficient as Python, as it is implemented in Java.

The final language being considered is C++. Developed in the 1980s as an extension to C, the language offers the flexibility of both writing procedural and object-oriented programs. There are a number of drawbacks as to why it is unsuitable for adoption in this project. Firstly, there is no standard GUI framework in C++. This means a third party package would have to be sourced, bringing about an increased learning curve before being able to make full use. There is also no standard support for regular expressions, which as detailed in the criteria, is an essential feature to this system.

3.1.3 Final Choice of Programming Languages

The programming languages which will be adopted for this project are Python and Java. Python will be used to develop the core logic of the system. The main reasons for choosing Python is that as desired in the criteria, the language is adequately scalable, offers support for regular expressions and allows for rapid development to take place. The NLP package 'NLTK', written for Python is also of great advantage and should help keep the time-scale for implementation on track.

Not having any experience in the use of GUI in Python, the writer also feels that Java should be adopted for developing the front-end. Drawing from previous experience, the GUI can be rapidly developed using Java and through initial research, a sufficient level of integration with Python is also supported.

3.2 Choice of Software Methodologies

3.2.1 Types of Methodology

Methodology	Summary	Applicability
SDLC (Software Development Life Cycle	Original software development life cycle fully described by Daniels and Yates, dates back to 1971. Idea is to follow through a number of stages sequentially. Each stage must be completed before proceeding to the next stage. Documentation is produced at the end of each stage allowing stakeholders to assess the development of the system (Avison and Fitzgerald, 1995).	Due to the complexities and time scale given for this project, a single iteration is the most feasible strategy. A single iteration was the original intention in using SDLC. With a large scope for future development, modified or extended requirements would require either further iterations of the SDLC or adoption of a different methodology.
Prototyping	*"A prototype is an approximation of a type that exhibits the essential features of the final version of that type"* (Avison and Fitzgerald, 1995). Appropriate when user requirements are not fully determined from the outset and where constant end-user involvement and evaluation is required to identify the requirements. One problem with prototyping, is knowing *"where the iterative process*	At the end of this project, a major deliverable is to have implemented a live system meeting the minimum plus any additional requirements (extensions). At the end of prototyping, the final prototype is not the 'live', fully operational system. Additionally, user requirements and scope for development are already fully determined after conducting background research. The iterative

	of revision stops and the development or delivery of the production system begins" (Avgerou and Cornford, 1998).	approach of prototyping is unfeasible for this type of project, whose complexity within in the implementation phase will only allow for a single iteration to be performed in the permitted timescale.
Spiral Model	*"Build a prototype using the Waterfall model, then revise the objectives as necessary, and build a new prototype"* (Raccoon, 1996). *"The major distinguishing feature of the spiral model is that it creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process"* (Boehm, 1988).	Since this model is aimed at large-scale projects, placing emphasis on evaluation of risks at the end of each iteration, this is not practical for the purposes of this project whose life cycle and time scale are relatively small. The iterative nature, just like which can be adopted in prototyping is also unfeasible since the nature of the implementation will only allow for a single iteration.

In evaluating the preceding methodologies, the writer chose to adopt the SDLC model. Although not all the stages are appropriate, the single iteration and the production of documentation at the end of each stage, make this the most suitable method. Having previous experience with this model is additionally helpful to minimising the learning curve involved in adoption.

3.2.2 Modifications of SDLC Model

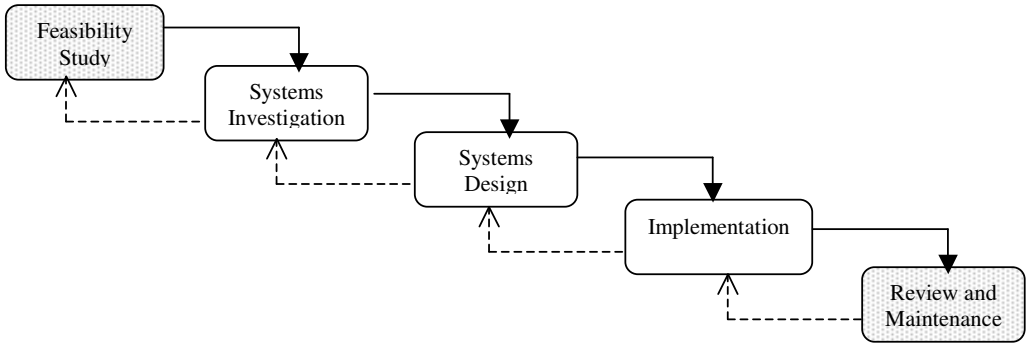


Figure 3.1: SDLC Model. Stages 'greyed-out' will not be included in the life cycle of this project.

SDLC, being the original life cycle model from which many methodologies have developed from, is flexible allowing modifications to be made as to which stages to be used and the content within each stage. The stages that will be used and modified for the purpose of completing this project are, 'Systems Investigation/Requirements Analysis', 'Systems Design' and 'Implementation'. The first stage, 'Feasibility Study' is not necessary since this focuses upon justifying the development of a new system to replace an older *system*. Since this project is developing a completely new type of system, no such comparisons and justifications can be made. The final stage 'Review and Maintenance' is also irrelevant for inclusion since it is aimed for systems, which constantly require changing after the system goes operational.

The Systems Investigation/Requirements Analysis phase will be adopted to elicit the functional and non-functional requirements of the system (see Appendix. E). These requirements will be based upon the previously conducted background reading into the domain of NLP. The set of requirements will fully cover the minimum requirements of this project along with any implemented additional requirements. The second phase used in this project is Systems Design. During this stage, the main system processes will be identified (see Appendix F) and the user interface will be designed (section 3.4). The final stage to be taken from the SDLC model is the implementation phase. Section 4 (Implementation), fully documents the actions taken to develop the system. The SDLC model includes testing within the implementation section. An interpretation of the results of all quantitative tests as well as a qualitative analysis is given in section 5 (Testing and Evaluation).

3.3 Choice of NLP Methods

From the research conducted during the background reading, two main methods need to be adopted for the implementation. Information retrieval is required to determine the key terms which best characterise a particular source document. Separately, the information retrieval method requires another method for constructing both the inverted index and also the source document's query terms. There is a choice of two techniques identified in the background reading section (2) for implementing the latter method. A choice has to be made between 'Collocations' and 'Term Extraction'.

For the purposes of this project, the words/phrases of interest are Noun Phrases (see section 4.2.3 for details on the nature of Noun Phrases used in this project). Noun Phrases are identified to discover the main concepts (topics) of an underlying document (or corpora).

Using the statistical method of Collocations to find Noun Phrases would require finding words with a high mutual information value i.e. those words that co-occur (possibly within a given window) together more frequently than they do separately. Words having high mutual value are mostly compound Noun Phrases. There are two drawbacks in adopting Collocations. The co-occurrence nature of Collocations limits identified phrases to bigrams. This is inadequate for finding index terms consisting of either a single word or of a large quantity of words. Another problem is that phrases of high mutual value are not always Noun Phrases; there is always a certain amount of noise. Although this is unavoidable, the noise of the identified phrases can be minimised through the use of a cut-off point on the mutual information value.

Term Extraction allows more effective filtering on the identification of certain compound phrases. Through the use of POS tagging and chunking, terms to be extracted can be limited to purely Noun Phrases. These terms can be either single words or phrases of an unrestricted length. For these reasons, term extraction will be used for determining the index terms and the query expression.

Information Retrieval will be adopted to find those terms from the query expression (the presentation source document), that uniquely describe the source document's contents from all the index terms (the scientific articles corpus). The method to be used for comparison will be the TF.IDF measure (described in section 2.2.3.4, Term Weighting). Although there are many statistical measures for performing this comparison, the TF.IDF scheme is among the most popular adopted methods in IR and is suitable for this project, being relatively straightforward to implement and offering an adequate level of accuracy.

3.4 User Interface Design

The interaction style that will be used in this project is 'Menu Selection'. Although 'Command Language' could also have been adopted, the 'Menu Selection' style is more suitable for catering for the needs of novices and experts alike. With a command line interface, novices must learn the correct syntax for any new tasks not previously accomplished or recalled. With the 'Menu Selection' interface, *"there is a clear structure to decision making, since all possible choices are presented at one time"* (Shneiderman and Plaisant, 2005). Popular existing presentation packages all use a Menu Selection Interface and so should reduce the learning curve required to become familiar with the look and feel of this system as well as its functionality.

When designing a user interface, research and the adoption of guidelines is essential to ensuring a user's experience of the system is an enjoyable one, behaving and functioning in a predictable manner. Shneiderman and Nielsen, both developed separate sets of heuristics, which offer pointers to producing 'user-friendly' interfaces. Shneiderman's 'Eight Golden Rules of Interface Design' (Shneiderman and Plaisant, 2005) and Nielsen's (1994) 'Ten Usability Heuristics' will now be adapted for the purposes of devising a set of guidelines by which the user interface of this project will follow. Throughout the implementation of the UI, regular reviews will take place to ensure these guidelines are followed and maintained.

- **Strive for Consistency** – In designing the user interface, consistency in the layout and functionality of the system should be maintained throughout. The terminology embedded in the menus, should correlate to the terminology used in titles of corresponding dialog boxes. A single colour scheme should be defined and adopted throughout.

- **Cater for Universal Usability** – The system must be flexible to cater for the need of both novice and expert users. By following the subsequent guideline, 'Standards', users who have had previous experience with other presentation packages, should find the layout of the system easier to learn and use. Since the target audience is conference presenters, it is assumed that the majority of users will have had some previous experience with one or more presentation packages. Expert users will be catered for by providing keyboard shortcuts for some of the regularly performed tasks.

- **Offer informative feedback** - *"For every user action, there should be system feedback"* (Shneiderman and Plaisant, 2005). In the case of all tasks supported by the system, feedback in the form of dialogue boxes will be given at appropriate points to provide confirmation of the current state of the system.

- **Prevent Errors** - *"As much as possible, design the system such that users cannot make serious errors"* (Shneiderman and Plaisant, 2005). Any errors a user makes should be detected by the interface and appropriate feedback should be given. *"Erroneous Actions should leave the system state unchanged, or the interface should give instructions about restoring the state"* (Shneiderman and Plaisant, 2005).

- **Permit easy reversal of actions** - *"As much as possible, actions should be reversible"* (Shneiderman and Plaisant, 2005). For appropriate tasks, functionality to allow a user to reverse the previously supported action shall be supported.

- **Standards** – As far as possible, in designing the UI, application and platform conventions shall be followed. The layout of the system will be similar to existing presentation packages. The functionality of the 'text editing' part of the system, will obviously be minimal in comparison to other presentation packages. This is to be expected since the system is not meant to be emulating a presentation package, but merely acting as an intermediary between extracting the relevant text from a source document and after exporting to another presentation package, adding aesthetics and additional visual content.

- **Help and Documentation** - A user manual shall be written for reference in assisting users to complete all supported tasks (Appendix H).

Section 4: Implementation

Having determined the design considerations in the previous section, the implementation section applies these design decisions, detailing the main development areas involved in implementing the system. This section is spilt into the parts of 'Corpus Collection', 'Key-word and Term Extraction', 'Slide Formatting' and 'User Interface'.

In the 'Corpus Collection', the exact usage of both the tagged corpora and the untagged scientific articles corpus is given. A description of the untagged corpus is given in section 2.1.2 (Corpus of Scientific Articles).

The proceeding part, 'Key-Word and Term Extraction' is the core function of this system, which will be explained in some depth of detail. The implementation of this method will draw upon the understandings of both 'Term Extraction' (section 2.2.2) and 'Information Retrieval' (section 2.2.3) gained during the background reading section. Knowledge gained in the second semester through DB32 (Technologies for Knowledge Management) will help to further reinforce the writer's understanding in Information Retrieval. During the design phase when choosing a selection of target languages, the use of regular expressions was identified as fundamental to development. Regular expressions were covered during the SY23 module (Internet Systems Technologies), providing valuable experience with which to apply in developing this method.

The Slide Formatting Section explains the implemented sectionalising method for segmenting the key terms identified from the 'Term Extraction' method. The UI will combine the 'Key Term' and 'Slide Formatting' methods to return a presentation containing a number of slides, each having a heading and a list of key terms.

The final part of the implementation section, 'User Interface', demonstrates how the front-end was constructed taking into account the framework/guidelines formulated in section 3.4. (User Interface Design).

4.1 Corpus Collection

For coherency reasons, before discussing in detail the various corpora used in the solution, a set of alias names will first be defined. When referring to the collection of all scientific articles, the alias CorpusSci will be used. This corpus is then split into the set of training scientific articles with which to perform tagging on (CorpusSciTR) and the set of scientific articles on which the testing section of the report is based (CorpusSciTest). The manually tagged corpus used for training the POS tagger is referred to as CorpusTagTr

The implementation of the tool for semi-automating the process of generating slides, requires the use of two distinct corpora. As the method chosen for extracting the key terms from documents involves adding

Part Of Speech tags to the CorpusSciTR, a set of manually tagged corpora (CorpusTagTr) is required for training a POS tagger. A tagged set of corpora is necessary to allow the system to subsequently tag the CorpusSciTR as part of the process of extracting key terms (see section 4.2). To ensure the system is robust enough to handle different domains with only minor modifications to the code, the CorpusTagTr must encompass many different domains and be as large as the system can handle before performance becomes an issue.

As discussed in section 2.1.2 (Corpus of Scientific Articles), a corpus collection of scientific articles is separately required for both compiling the set of training data (CorpusSciTR) from which the term extraction system will generate the inverted index and also for carrying out the evaluation (CorpusSciTest) of the system against documents not used in the training set. To fulfil these two distinct requirements, the corpus needs to be segregated in such a way as to give the system sufficient training material to construct a representative inverted index, while leaving enough documents to be able to carry out a thorough evaluation.

A number of design considerations concerning the size of the CorpusSci (number and length of files) and the domain(s) of the CorpusTagTr needs to be decided upon at this point. The following two sections discuss in detail the exact usage of the CorpusTagTr and the CorpusSci for the purposes of implementing the application.

4.1.1 Tagged Corpora

The main consideration for the CorpusTagTr, involved deciding upon which corpus collections to use, in order to elicit a representative sample of English, covering a number of different domains. Ideally, the best corpus with which to train a POS tagger on would be one from the scientific domain itself. Since no such corpus was available at the time of developing the solution, alternative ‘English-focussed’ corpora had to be found. Having previously decided in the design stage upon the use of the python programming language utilising the NLTK package for developing the term extraction system, the scope from which to choose a representative corpora was limited. Only four different corpora in the format able to be read into the NLTK package were at the writer’s disposal at the time of writing. The choice of corpora from which a selection had to be decided upon was an Air Traffic Information System, The Brown Corpus, The Switchboard Corpus and a collection of Wall Street Journals.

The Brown corpus was the most obvious choice for inclusion into the tagged corpora collection as it was compiled specifically to offer a collection of documents, which aim to be representative across a multitude of domains, capturing the essence of American English. In the selection process, a representative sample was obtained not only by choosing documents from different fields, but also the number of documents in each field was chosen in a weighted fashion towards those domains encompassing a wide range of issues pertinent to the American society at the time of construction (Francis and Kucera, 1979). The entire corpus consists of over 1 million words, which for the purposes of training the POS tagger (Brill

Tagger), offers a good basis from which a large proportion of words in the CorpusSciTR can be assigned tags.

As the system is primarily aimed for use in conferences, the content of the presentation in most cases is expected to be of a technical nature. From the remaining corpora, The Wall Street Journals contains documents, which are all highly focussed to a specific technical domain. Although the domains covered by the WSJ are mostly different from the domain of the CorpusSci, the additional 3 million words embodied within the WSJ will help improve the vocabulary with which to train the POS tagger. The final corpus chosen to be included in the tagged corpora collection is the Switchboard Corpus. On first inspection, the relevance of this corpus may be questionable given that it contains spoken conversational English comprising slang words and incomplete sentences. Through initial testing however, training on a sub-selection of this corpus and tagging an untagged document (from the CorpusSciTR), the Switchboard Corpus managed to tag a significant number of words not covered by either The Brown Corpus or The Wall Street Journals. This may be due to the fact that in conversational English, a wide range of topics and domains are discussed and also because there is a tendency in spoken conversations to use shorter words to encapsulate the same semantics as what may be written in the documented equivalent. The only other corpus available, a collection of dialogues from an Air Traffic Information System (American), does not facilitate much improvement in tagging scientific documents, since it focuses on a very narrow field incorporating the geography of America with common expressions of automated travel information systems. The total number of words in all three corpora is over 7 million words. This gives a training corpus (CorpusTagTr), which is large enough to tag a high proportion of documents in a range of domains. The limiting factor now becomes one of determining what proportion of the total words can be handled by the NLTK package before the response/computational time becomes infeasible for the purposes of this project.

The task of training the POS tagger (Unigram and Brill taggers) on the previously chosen CorpusTagTr, ultimately determined the number of words, which could be feasibly handled. The operation performed by the 'Brill Tagger', involves scanning through the CorpusTagTr's POS tags assigned to words/punctuation marks and developing a set of transformation rules from which to minimise tagging errors. This is computationally intensive and after several trials, a cut-off point of 25 items (files) per tagged directory was considered optimal. This corresponds to around 2 million tokens and the operation on this size takes about 10 hours to complete. For this reason, the model and rules resulting from training the Brill tagger is stored and compressed into a file for subsequent loading, taking only 1-2 minutes. The reason for specifying a cut-off point in terms of tagged items (files) per directory, stems from the fact that the system where possible tries to take measures to ensure domain independence. Capturing a certain number of tokens covering an assortment of domains widens the domain scope from which the sample will be compiled.

4.1.2 Untagged Corpus of Scientific Articles

The design consideration in utilising the CorpusSci is one of trade off, between using sufficient documents for the CorpusSciTR to accurately aid in generating a list of key terms, against having sufficient testing documents (CorpusSciTest) to evaluate the precision of the generated key terms.

Knowing that a number of operations were likely to be computationally intensive, after implementing key term extraction so far as being able to create the inverted index for the CorpusSciTR, tests were carried out using all the articles in varying combinations of directories. Out of all the operations performed in creating the inverted index, the operation of identifying noun phrases was found to take the longest duration. A cut-off point involving utilising three out of the four directories with which the articles are located was determined. The three directories in total contain almost 650,000 tokens and this takes around 12 hours to find the noun phrases through the NLTK package. As the inverted index will not need to be re-created every time the application is executed, the final index is saved to a file, allowing loading on subsequent executions to take no more than 10 seconds. The total number of files in the three directories is 186. After ignoring articles not meeting the criteria mentioned in the problem definition (section 1.1), there are 160 articles or 630,000 tokens in the CorpusSciTR. This leaves one folder, aptly named ‘unseen’, containing a further 168 articles. This will be more than enough for the CorpusSciTest, allowing testing to be performed along the implementation cycle as the key-term extraction solution is optimised.

4.2 Key-Word and Term Extraction

This section of the report is split up into the main operations and steps required to decide for a given unseen XML document (taken from CorpusSciTest), which terms characterise it best. Unless otherwise stated, the implementation stages apply equally to both the CorpusSciTR and the unseen source document.

4.2.1 STAGE1: Strip XML to obtain Article’s Content

Before a tagger can tag a particular article, the XML tags and the references located within the document must be removed leaving only the document’s content. As the NLTK library will be used for implementing most of the proceeding stages, the article’s content must then be fed into a data structure supported by the library. During the explanation given in the rest of this section, it may be useful to refer to the sample XML scientific article located in Appendix D, as well as the associated DTD Scheme (Appendix C).

Stripping the document of its XML tags and returning just its content, involves using a number of regular expressions. Firstly, a regular expression was devised to find the start and end of the document’s body i.e. looking for the sentence tags. All matches except the last (which returns everything to the end of the document including references) were then re-combined.

Another regular expression handled removal of words/phrases within the document's body embedded within a pair of XML tags. Such examples of words/phrases not needed to be included when determining the key terms of a document are image names and citations. In the case of citations, the names of a number of authors may be referenced in a particular document, but unless a particular name occurs very frequently, none can be considered to be candidate terms uniquely identifying the document. Any remaining XML tags are then deleted with a separate array of regular expressions.

Having got the plaintext content of the document, punctuation marks are then identified and in each instance, a white space is added. This is an essential step to ensuring the tagger is given every chance to separately identify the preceding and proceeding word from the punctuation mark.

A final set of regular expressions performs cleaning of the remaining content, removing unnecessary white space and deleting several non-alphanumeric characters. The content at this stage is now ready to be processed and read into the NLTK library. To do this, the string containing the content of the document must be converted into a set of tokens. This is achieved by using the white space tokenizer method of NLTK to create a new token for each word (between pairs of white spaces).

A token, is the data structure used by NLTK to store words/phrases and additional associated information. A token typically consists of a 'text' field where the actual word/phrase is placed, a 'location' field holding the offset of the term from the beginning of the document and a 'POS' field giving the token's Part Of Speech tag. Additionally, any user-defined property may be added to each token. Tokens can be hierarchical as one of the properties of all tokens is 'Subtokens'. A group of tokens relating to each other may be individually called through their common parent's 'Subtoken' variable.

At the time of converting from a string to a set of tokens, an argument is passed to make sure the location offset of each word is stored in the token's 'LOC' property. Additionally, a user defined field 'FILE' is specified passing the filename with which the instance of a word was located. On completing the tokenisation of words and phrases and in adding the additional information to each token, the article(s) are now ready to be tagged.

4.2.2 STAGE2: Create Tagged CorpusSciTR/Source Document

Before a document can be tagged, a tagger needs to be trained on a tagged corpus (CorpusTagTr). The resulting model (transformation rules that are optimised on the training corpus) can then be applied to an unseen, untagged document from CorpusSciTest or the untagged training set CorpusSciTR.

Now that the CorpusSciTR has been tokenised, a few additional pre-processing operations can be performed to help maximise the proportion of all words that the system is able to assign POS tags. As python (and the NLTK library) is case sensitive, the first operation required was to convert all the tokens to lower case. This is essential both for performing comparisons and also in computing frequencies, ensuring

the same two words, formerly in a mixture of cases are treated by the program as being conceptually the same.

The next task in the tagging process requires the use of two taggers, a Unigram and a Brill tagger (supplied with the NLTK library). These taggers combined, are trained on the CorpusTagTr and result in assigning tags to the CorpusSciTR (or the source document). The Unigram tagger is a stochastic tagger, taking a tokenised tagged corpus and for each token (word type), computing the most frequently occurring POS tag. After training the Unigram tagger, the Brill Tagger works on top of the Unigram tagger to minimise tagging errors. The trained Unigram tagger is passed as an argument to the Brill tagger, which then proceeds to generate an ordered set of transformation rules from which to reduce both the number of untagged tokens and the amount of tagging errors for tokens having ambiguous possible POS assignments. An example of such a transformation rule is that given the word 'work', tagged by the Unigram tagger with the most frequently occurring tag i.e. 'NN', a transformation rule may be found to replace all instances of the tag 'NN' (noun) with the tag 'VB' (verb), for all uses of 'work' preceded by the tag 'TO'. As demonstrated with the preceding rule, transformation rules can consist of both POS tags and actual words. The following sentence, 'I have to go to work/NN', after applying the previous rule, would alter to 'I have to go to work/VB'. In other contexts where 'work' appears, for example, 'This work/NN is full of surprises' would remain unaffected by the previous rule. When generating the transformation rules, a set of templates is required defining the scope with which the tagger will search to formulate these rules. The set of templates used in this implementation, limited the scope for finding rules, by allowing comparison of a given token to take place with no more than the two proceeding tokens and the adjacent preceding token. Having the tagger search any greater distances from a token is of little benefit since most useful rules are based upon comparing with the tags assigned to adjacent tokens. Additionally, the extra time required to search greater distances would severely compromise the size of the tagged corpus training set that could be computed beneath the previously determined cut-off point of 10 hours. A further parameter, limiting the amount of generated rules to 10, was defined also to comply with this time restriction.

After training the Brill tagger, the tagger's state was serialised, saved to a file so that subsequent executions of the program would not have to repeat this time consuming operation. Due to the amount of tokens processed through the Brill tagger from the CorpusTagTr, the resulting file required compression. This is true of many of the subsequent files, which in some cases are holding a mixture of several million characters and words. This is an important consideration as potentially, this application may be used in an environment where lecturers/presenters have a limited quota of hard disk space.

Before continuing to tag the CorpusSciTR, a measure of the tagger's accuracy was required. For finding the tagger's accuracy, the NLTK library provided a function for evaluating a given tagger against a tagged corpus. The function operates by taking the tagged corpus, stripping-off its POS tags, re-tagging the collection through a tagger and then comparing the result with the original tagged corpus. The accuracy of the Brill Tagger was computed using the same corpus as the one used to actually train the Brill Tagger on. The computed accuracy will of course be lower if calculated on the tagged CorpusSciTR. After running this

function on the Brill tagger, trained on the CorpusTagTr, an accuracy of 0.95 or 95% (equivalence) was reported. In the context of this project, the accuracy of the tagger was deemed sufficient and as such, the next operation of tagging the CorpusSciTR proceeded. This operation took a shorter time to compute, though the result was again saved to a file for faster retrieval on subsequent executions.

At this point, finding the percentage of remaining untagged tokens in CorpusSciTR was necessary to know whether further processing would be required to reduce the proportion to an acceptable level. The percentage of remaining untagged tokens was found to be 15%. Upon further investigation, there was a considerable proportional of unambiguous, untagged words, which although weren't found within the tagged corpus or tagged as a result of the transformation rules, could be tagged by use of a dictionary. The 'Wordnet' dictionary already installed on Leeds University's School of Computing machines was utilised to assist in reducing the number of untagged words.

Four separate iterations making use of the Wordnet dictionary were necessary to reduce the percentage of untagged tokens. On the first iteration, all unambiguous words were looked up in Wordnet and where a matching word was found, its associated POS tag was assigned to the token. On the second iteration of the remaining untagged tokens, the morphological base feature of Wordnet was determined for each word and the resulting 'bases' were then looked up in Wordnet's dictionary. The third iteration involved, finding hyphenated words and looking up in Wordnet, the part of the word preceding the hyphen. Separately, the word occurring after the hyphen was looked up and if a match was obtained, this was appended as a new word in the list with its associated POS tag. The final iteration identified the remaining untagged hyphenated words, removed the hyphen and merged the two words together. The result in each case was looked up in Wordnet.

A final aid to reduce the number of untagged tokens involved manually supplying a set of patterns and words with their associated POS tags. Any matching tokens to either the patterns or words were assigned the supplied POS tag.

The percentage of untagged tokens after applying all the above operations was reduced to 8%. To make any further reductions would require more exhaustive operations, which in the end would be more detrimental to accuracy beyond the little reduction, which would be gained. The final set of tagged words was again saved to file for subsequent retrieval.

4.2.3 STAGE3: Generate Noun Phrase List

Turning the set of CorpusSciTR tagged tokens into a list of noun phrases was achieved through using what is known as a chunker. The chunker supplied with NLTK, permitted regular expressions to be formulated, searching for a certain sequence of POS tags within the tagged list. When a matching sequence is found, parenthesis is added at the start and end of the phrase, indicating a chunked item. The particular regular expression used for identifying noun phrases involved looking for any sequence of adjacent tags

made up from determiners (DT), adjectives (JJ) and all noun derivations (NN*). Only base noun phrases such as 'the logic grammar paradigm' are discovered. Other phrasal forms like the prepositional phrase 'He went in the building' and clauses such as the relative clause 'The car I bought was on sale' are not considered. The chunk operation when performed on the CorpusSciTR set proved to be extremely time consuming, taking 12 hours to process the 630,000 tokens. The generated noun phrase list for both the source document and CorpusSciTest set was saved for faster retrieval.

4.2.4 STAGE4: Create Inverted Index and Source Document Term Frequency Index

The operation of creating an index for both the CorpusSciTR and the source document is sufficiently different to be discussed in detail separately. The following section will begin documenting the steps involved in obtaining the inverted index from the CorpusSciTR, followed by the steps completed to produce the term frequency index for the source document.

4.2.4.1 Inverted Index

Before construction of the inverted index could be performed, a pre-processing step was necessary to obtain a set of distinct terms. This required taking the list of identified noun phrases and removing any duplicates. The frequency distribution class offered by NLTK performed this exact operation by taking the list of noun phrases and grouping identical phrases, returning a distinct list.

For each term in the distinct list, the documents it occurs in and the frequency in each document was then computed. This was done by taking each distinct term and comparing against the original (non-distinct) set of noun phrases. Where two terms between the two sets matched, two arrays storing the files and their associated frequencies were updated. This was a three-pronged process of determining for a given match, if any files had already been identified for the term, if the file already existed in the files array for that term, or if the files array of the term already contained some files, but not the file in question. In the case of no files existing for a given term, the filename was appended to the file array and a new frequency of one was appended to the frequency array. Where a filename already existed, its associated frequency was simply incremented by one. In the final scenario where the file array was not empty, but did not contain the file/document in question, the filename was appended to the file array and its associated frequency set to one. After each term in the distinct list had been compared with all terms in the non-distinct list, the term's document frequency was assigned based upon the length/size of the array holding all the filenames. Additionally, the 'documents' tag of each token was assigned with the actual names of all the files in which the term appeared. Below is an example of the format of the stored inverted index:

rhetorical structure/ 9411023.sent(5), 9502018.sent(4), /2
text segmentation/000383.sent(8), 9812005.sent(1), /2

The format of the inverted index is: 'term / documents term appears in with frequencies in each case / total number of documents term appears in'. This index is sorted in descending order of document count. The final count for the number of distinct index terms is 17,700. Although this is a considerable size smaller than the original number of identified terms due to focussing only on the noun phrase type, since each term can be said to characterise a certain aspect within the domain of Science, the resulting list is sizable for capturing the essence of the domain in question.

4.2.4.2 Term frequency Index

The first stage in creating the term frequency of each term, like for the CorpusSciTR, involved using the frequency distribution method to remove duplicates and return a list of distinct noun phrases. This method also permitted the frequency of each distinct term to be computed. To support greater precision while utilising domain independent techniques, a stop list was applied to remove common words/phrases from the resulting distinct list.

Each token's 'text' and 'term frequency' (as elicited from the frequency distribution function) fields were saved into the source document's term frequency file. The range of distinct terms identified for all unseen documents is 100-300.

4.2.5 STAGE5: Rank Terms in Source Document using TF.IDF

The ranking of terms starts by reading in the previously saved inverted index and source document term frequency index. In processing each document term, a regular expression was used to identify whether the token in question was a multi-phrase unit or a single word. If the term was found to be multi-phrase, the term frequency variable for the token was increased by a factor of three. If on the other hand the phrase was a single word, then the term frequency variable was simply assigned the term frequency of the token in question. The reason for increasing the weighting of multi-phrase units is simply that phrases within a technical field usually contain more semantically meaningful concepts than can be drawn from individual words relating to the domain being depicted. Increasing the weight by a factor of 3 was an ad-hoc step based upon testing and reviewing the results for a small number of documents. In future, a more accurate approach should be adopted to say, compute some frequency measure to determine weighting.

Having determined the TF of a particular term, the DF belonging to that term within the inverted index had to be extracted. Once the TF and DF had been obtained, the term's TF.IDF score was then computed. As explained in the Information Retrieval (section 2.2.3) part of the background reading section, this was computed as: TF.IDF = tf * log(N/df) where 'N' is the total number of documents in the training set i.e. 160. With all the TF.IDF scores computed, the list of document tokens (terms) were re-ordered in order of highest TF.IDF score descending. A user-defined variable, determines the number of key terms, which will be both saved to file and displayed to the user.

4.3 Slide Formatting

Having extracted a sub-selection of key terms characterising a particular source document, the focus in the implementation shifted towards developing a method by which to construct a set of presentation slides. Although the original plan at this stage was to base the slide segmentation around the researched field of TextTiling, due to time constraints and with the emphasis of this project being placed on implementing a method for term extraction, a more basic 'sectionalising' technique was adopted. The rest of this section offers a detailed explanation into the functionality of the implemented method as used by the system.

Prior to performing segmentation on the key terms, a method was required for generating a title slide i.e. the first slide of any presentation. On the basis of the information afforded by the use of XML in the source document as well as the through prior knowledge and research, the desired information to appear on the title slide was determined to be the title and the author(s) of the article. Two regular expressions handled searching for the title (<TITLE>) and author(s) (<AUTHOR>) of the document. As documents can have more than one and indeed many authors, a set of conditional statements were used to generate appropriate output. With one author, the output would just return the author's name. Documents with exactly two authors would return in the format of 'Martin Shaw and James Thompson'. For three or more authors, only the first identified author would be mentioned taking the format of 'Larry Patterson et al.'. The extracted title and formatted author(s) is returned by this method and passed onto the main method, dealing with segmenting the key terms of the source document.

The aim of the implemented segmentation method is to generate a list of slide content. Each slide contains a heading as found in the document and all previously identified key terms, which appear within the boundaries of this heading. The initial operation requires finding and storing the location of all (sub-)headings in the source document. For this, the whitespace tokenizer class of the NLTK package was utilised to split the document's text and add location information to each identified token. All tokens are then searched with a regular expression to find those, which are headings i.e. contain the 'header' tag. Each identified header together with its location was stored in a 2D array.

In order to perform the comparison of key terms with headings, the location of all instances of each key term in the source document had to be determined. Each instance discovered was stored together with its location in a 2D array.

The comparison between the two arrays holding headers and instances of key terms could now take place. Each comparison required comparing the location of an instance of a key term, with all headers. A variable handled storing the position of the closest header. When all headers had been compared, the header number of closest header was appended as an additional attribute to the array storing term instances.

After determining the location of all instances, a new 2D array ('lectureterms') was created to hold the final content of each slide. Each element in this array held a header taken from the headers array and all key term instances in the instances array flagged with the appropriate header number.

The final stage was to save the slide content to file. As previously discussed, each slideshow begins with an introductory slide containing the title and author(s). The content of the title slide as determined by the first method was the first text appended to file. Then the content's of all elements in 'lectureterms' was added. Before outputting each element (slide) to the file, a special character sequence '////' had to be outputted to mark the beginning of a new slide. An element is then written to file by taking its first attribute i.e. header and appending two new line characters. Each of the remaining element's attributes (key-terms) are all written to file on separate lines.

4.4 User Interface

The final phase in the implementation was to develop the front-end/UI of the system to interact between the user and the underlying functionality. Before implementation began, based upon the research conducted during the design stage (section 3.4), a template layout had to be determined by which all views of the system followed.

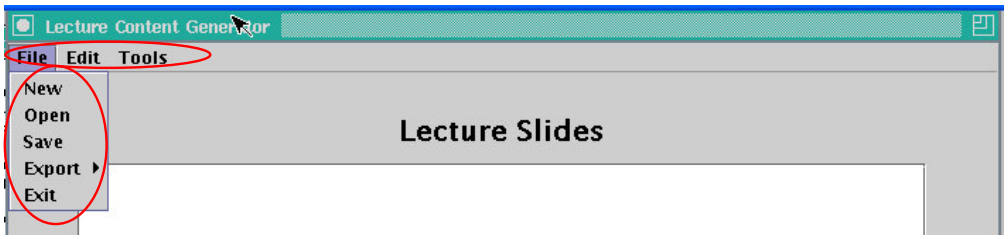


Figure 4.1: The system's menu interface

The first consideration was the placement of the Menu Bar. Following the 'Standards' guideline in section 3.4, as most other presentation packages use a horizontal bar at the top of the window, this is the position in which the menu bar is also placed in this system. There are three submenus in this bar, 'File', 'Edit' and 'Tools'. The names/terminology used throughout the menus and options, again follow standard application conventions as suggested in the 'Standards' guideline. The order of both the menus and options in each menu as can be seen in figure 4.1, was also chosen in such a way as to follow 'de facto' application conventions.

The placement of the remaining GUI components for the main view of the system also had to be determined in accordance with the guidelines developed during the design phase (section 3.4). The background colour and the menu colour adopted was the default 'grey'. This again conforms to application conventions and ensures startling, vibrant colours do not unnecessarily distract the user. The main

components to be positioned at this point were the ‘Lecture Slides’ text editing box (showing the content of each lecture slide), the ‘links to further research’ text box, and the navigation buttons for moving around the slides.

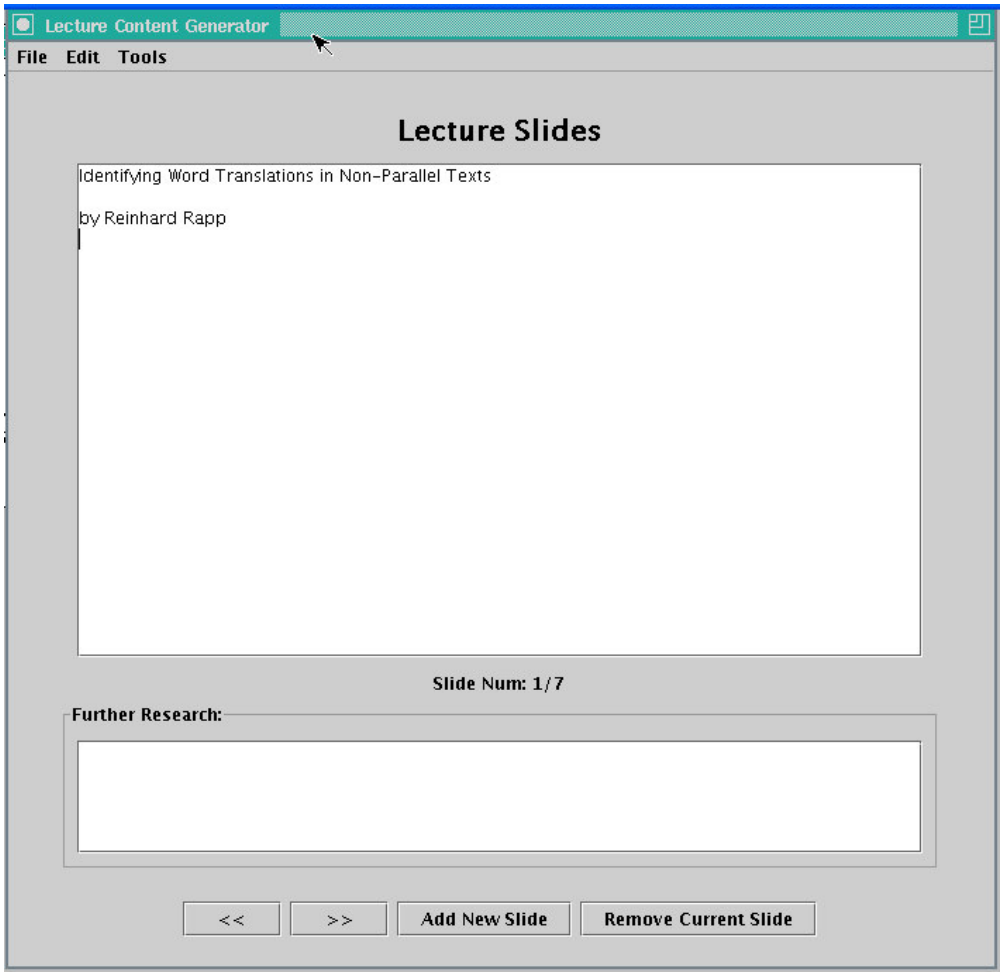


Figure 4.2: Front-end of Slideshow Generator

Since the main focus of this application concerns the lecture content as opposed to offering pointers to further research material relating to a particular source document, the lecture text box was given a prominent position, placed just below the menu bar. In order that the operation of editing each slide of a presentation does not require a user to onerously scroll using both the horizontal and vertical scroll bars, the dimensions of the text area were set, to take up the entire width and half the height of the window. Directly above the ‘lecture box’ is the title ‘Lecture Slides’ to further clarify to the user the meaning and purpose of the text area. Below the text box is another label, showing the current slide number, which a user is viewing as well the total number of slides in the presentation. This is essential not only as it follows application conventions, but also so as the user can know the current state of the system i.e. the slide, which they are viewing. The inclusion and structure of this label directly correlates to the guideline ‘Offer Informative Feedback’ proposed in the design stage (3.4).

The further research box was positioned below the lecture slides text box and associated labels, taking up the entire width of the window, but only a small proportion of the window's height. Since URLs can be lengthy, it is necessary that the window does take up the full width. In the 'Tools' menu, an option exists to allow the user to alter the number of URLs that are returned. The maximum number that can be returned, is no greater than 10. For this reason, the height of the box did not have to be particularly large. The functionality behind this box takes the top 3 identified key-terms (from the term extraction method) and using the Google API, performs a standard 'conjunction' query, filtering to only academic institution domains. Although the system is primarily targeted at conference presenters who are only interested in presenting their own new material, lecturers may also make use of this system and would benefit from being offered pointers to relevant resources.

The final set of GUI components to be positioned, were the navigation buttons. Since the height the 'further research' box took up was small, the buttons were placed directly below at the bottom of the window. Even though the buttons have no impact on the research box, the placement of the buttons at the bottom to allow the user to interact with the main 'lecture slide' text box, follows application standards of providing a clear distinction between the areas where content is generated/edited and the areas where navigation and tasks/operations can be executed via menus or buttons. Figure 4.2, shows the final placement of all components visible throughout all performable tasks of the system.

After deciding and implementing the layout of the system, additional HCI considerations had to be taken into account when linking the underlying functionality with the front-end. Throughout implementing the UI, provision had to be made for the varying experience of users. As well listing all the functionality in the 'Menu Bar' for novices and experts to use alike, a few common keyboard shortcuts were supported to help increase the performance by which experts can carry out the task of editing lecture content. This provision stems from the 'Cater for Universal Usability' guideline from section 3.4. The specific keyboard shortcuts supported, allow for cutting, copying and pasting text held in the 'lecture slide' text box.

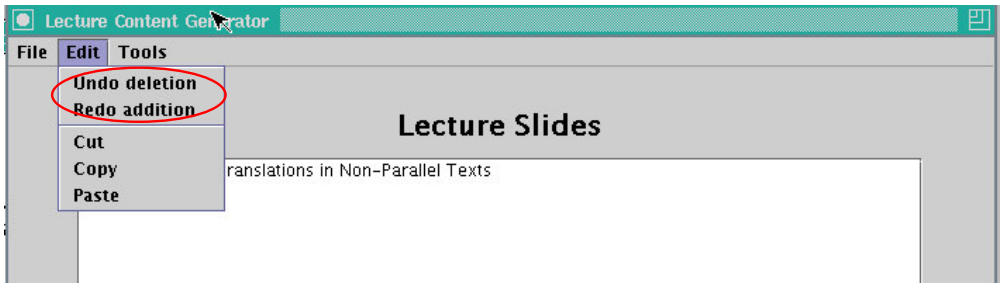


Figure 4.3: Editing options supported by the system's menu system

As with any text-editing application, all users are bound to make mistakes from time to time. The implemented 'edit' menu assists the user in reversing previous actions performed with regard to the slide content. The inclusion of 'undo' and 'redo' actions was motivated by the 'Permit easy reversal of actions' guideline of section 3.4, giving users the confidence to know any mistakes they make can be reversed.

Additionally, before executing one of the two commands, the user is made aware what exactly they will be undoing/redoing (figure 4.3) e.g. a paste operation, a deletion, etc. This further assists the user in deciding whether to invoke an operation and is directly motivated from the 'Offer Informative Feedback' heuristic.

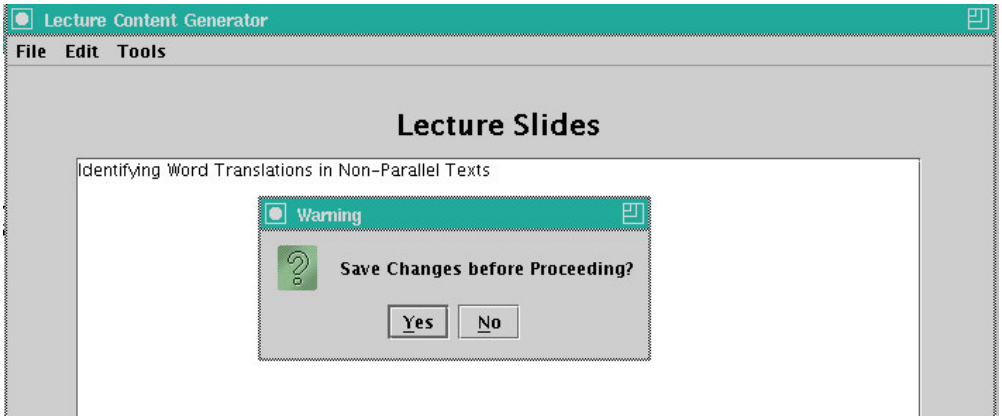


Figure 4.4: Text-Box prompt for saving slideshow before continuing to perform intended action.

In performing the 'New', 'Open', 'Save' and 'Exit' operations, careful preventative measures have been implemented to minimise the chance of a user making any serious errors. If at the point of a user selecting one of these operations (except the save option), the state of the 'Lecture' text area has changed since the last 'save operation', a dialogue box will give the option for a user to save their slideshow content (figure 4.4) before continuing to perform the intended operation.

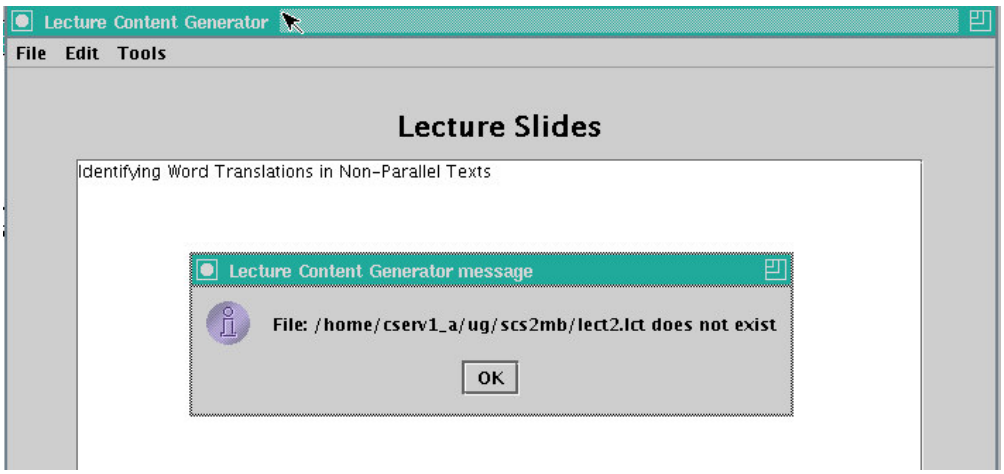


Fig. 4.5: Text-Box prompt informing user, the reason for failing to open/read file

In the case of the 'New' and 'Open' commands, where a file chooser box allows a user to enter a filename to open, feedback is given in the event of an invalid file being entered. As seen in figure 4.5, a dialogue box tells the user the nature of the error, along with the filename they attempted to open. These

error prevention and feedback measures arose from the ‘Prevent Errors’ heuristic found in section 3.4 (User Interface Design).

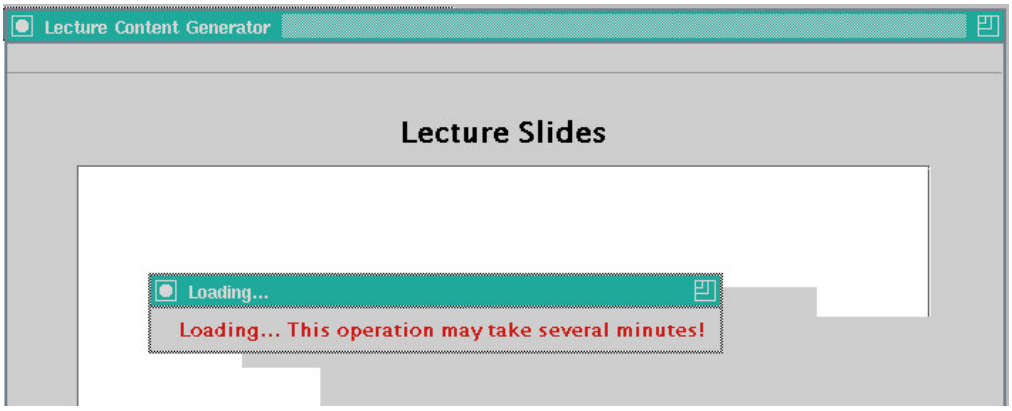


Figure 4.6: Loading Message Box. Brings to the user’s attention the fact that the operation is in progress and may take some time to complete.

As the time between the user invoking the ‘New’ operation and the visibility of the lecture content and further research links takes several minutes to complete, the user needs some idea that their action is being processed. To offer the user immediate feedback after executing the command, as seen in figure 4.6, a pop-up message box informs the user that their action is in progress and may take several minutes to complete. Upon completion, the message box disappears and the content is generated and displayed.

Section 5: Testing and Evaluation

Throughout the implementation of this project, intermediary tests were conducted to identify areas for optimising and improving the accuracy of the term extraction method. The testing and evaluation chapter of this report focuses entirely upon evaluating the quality/accuracy of the final build of the term extraction method. As a result of the emphasis in this project being based around developing and implementing this method, due to time constraints, there will be no section evaluating the user interface. This would require feedback from a number of potential users and would be infeasible given the timescale and focus of this project.

5.1 Testing

In order to carefully evaluate the implemented term extraction method, tests need to be performed to quantify the accuracy of the generated output. The testing section covered in this report is split into two parts, the 'human judge' test and the 'title and abstract' test. The human judge test, involves the use of real life conference presenters extracting key terms from several documents and comparing them to the output from the system. This test can be said to have some 'ecological validity', since the test is conducted by potential, real life users. Since the terms elicited from conference presenters will be subjective, a further, more objective test will be conducted.

The additional test will involve comparing the terms outputted by the system against the key terms that are present in the title and abstract. The results for the 'title and abstract' test should not be given the same importance as the 'human judge' test when evaluating the system. The abstract is only between 3-10 sentences long whereas the main text contains numerous paragraphs. As the term extraction system considers both the abstract and the main text, it is obviously more likely to find terms with a high TF.IDF score occurring in the main text than in only the abstract. In the 'human judge' test, key terms can be extracted from both the abstract and the main text.

It should be noted that the 'human judge' test was not part of the original planned tests for evaluating the term extraction method. During the research/background reading for this project, authors of the documents in the CorpusSciTest were emailed to attempt to retrieve corresponding slideshow presentations, accompanying their article(s). The idea was that given a scientific article and its corresponding slideshow presentation, key terms would be extracted from the slideshow in a fashion similar to the guidelines for extracting terms given to judges in the 'human judge' test (see Appendix I). This test would have had greater 'ecological validity' since the actual presentations as created by conference presenters would have been used in comparison to the generated key terms of this system. After a month of few responses, this idea for testing was abandoned. The main reasons as given in the few responses received were that many of the talks given from the material in CorpusSci, were 6 years or older. Presenters had either lost their electronic presentation or not originally produced it electronically i.e. just used OHP, whiteboards, etc… The 'Human Judge' test

was considered the next best alternative to determining the accuracy of the term extraction method offering the 'ecological validity' of at least involving conference presenters.

To conduct the two tests, one or more appropriate evaluation measures must be determined. As the field in which the term extraction method derives from is Information Retrieval, the evaluation measures of precision and recall are applicable to assessing the term extraction method. An understanding of these two measures was grasped during the DB32 module (Technologies for Knowledge Management). As mentioned in the evaluation section of the background reading into IR (section 2.2.3.5), precision measures the percentage of returned terms, which also appear in a separate compiled list. Applied to the two tests, precision would be determined by computing the percentage of terms generated by the system which also appear in either a list of all key terms in the title and abstract, or the list of terms extracted by a human judge. Recall, measures the percentage of all relevant terms, which also appear in a separate list of terms. The recall measure would be determined in the tests by calculating the percentage of all key terms present in either the title and abstract, or for the human judge test, the percentage of all key terms extracted by a human judge, which also appear in the key terms generated by the system. The reason for choosing to adopt precision is to demonstrate that the system's TF.IDF scheme, accurately ranks key-terms in order of those, which best characterise a particular document. Precision does not tell the complete picture however, since obtaining a high precision value does not give any indication as to the proportion of key terms identified from all possible key terms. A separate evaluation measure of recall is required to give this indication.

In the case of both compiling the list of all noun phrases in the title and abstract, and the list of key terms extracted by the human judge, clusters of associated words will be included in the list. For example, suppose a human judge had found the terms 'NLP' and 'Natural Language Processing' as both characterising a particular document, instead of listing them as two separate terms, they would be clustered as a single key term.

Both tests will be conducted on a sub selection of the CorpusSciTest identified in the corpus collection section (4.1). The specific scientific articles to be used from this corpus have not thus far been utilised in either training the system (separate set of documents used for this i.e. CorpusSciTR) or in intermediary tests throughout the implementation phase. The average precision and recall values determined from conducting the tests on these articles will form the basis with which to statistically evaluate the accuracy of the term extraction method.

5.1.1 Human Judges

The 'human judge' test involved asking conference presenters i.e. potential users, to look at fifteen different documents and in each case, to extract 10 noun phrases which best characterise that document. To ensure a human judge has a clear picture of the exact type of phrases, which should be identified, guidelines (Appendix I) were given in support of the documents. The guidelines give detailed

instructions in how to perform the test as well as the nature and structure of phrases from which key terms should be identified.

After a human judge has completed the test, the terms they identified are then compared against the top ten terms generated by the system for each corresponding document. Precision is calculated by finding the percentage of the top ten (ranked in descending order of TF.IDF score) key terms generated by the system against the ten (clusters of) key terms extracted by the human judge. The recall measure is computed by finding the percentage of key terms generated by the judge that also appear in the top ten key terms from the system output.

The following table shows the results for the two human judges' comparison against the term extraction output, generated by the system.

Human Judged Key Terms		P1		P2	
Document (.sent)	Group Size	precision	recall	precision	recall
9407001	10	0.3	0.3	0.2	0.2
9505004	10	0.2	0.2	0.1	0.1
9505007	10	0.1	0.1	0.3	0.3
9505008	10	0.2	0.2	0.2	0.2
9505009	10	0.1	0.1	0.1	0.1
9505010	10	0.2	0.2	0.2	0.2
9505011	10	0.4	0.3	0.4	0.4
9505024	10	0.3	0.3	0.5	0.4
9505025	10	0.3	0.3	0.2	0.2
9505028	10	0.2	0.2	0.3	0.3
9505030	10	0.2	0.2	0.1	0.1
9506025	10	0.1	0.1	0.2	0.2
9508005	10	0.2	0.2	0.3	0.3
9509001	10	0.4	0.4	0.4	0.3
9512003	10	0.4	0.4	0.5	0.5
Average	-	0.24	0.23	0.27	0.25

Overall Precision Average	0.25
Overall Recall Average	0.24

5.1.2 Title and Abstract

The 'title and abstract' test involved finding the percentage of key terms generated by the system that also featured in the corresponding identified key terms found in the title and abstract of an article. The list of all noun phrases occurring in the title and abstract was compiled by the writer identifying all instances of words/phrases which have the same part-of-speech structure, as outlined in the guidelines given to human judges in the previous test.

The precision measure was calculated by finding the percentage of the top ten key terms generated by the system, against the compiled list of all noun phrases found in the title and abstract of an article. The recall measure was determined by finding the percentage of all noun phrases of the title and abstract, which also appeared in the top ten key term results of the system.

The table below shows the results of performing the 'title and abstract' test on fifteen selected documents taken from CorpusSciTest.

Title and Abstract			
Document (.sent)	Group Size	precision	recall
9407001	24	0.4	0.17
9505004	26	0.2	0.04
9505007	15	0.2	0.13
9505008	14	0.2	0.14
9505009	19	0.1	0.05
9505010	15	0	0
9505011	13	0	0
9505024	21	0.1	0.05
9505025	19	0.2	0.11
9505028	20	0.3	0.15
9505030	17	0.1	0.06
9506025	23	0.4	0.17
9508005	19	0.2	0.11
9509001	32	0	0
9512003	20	0.2	0.1
Average	-	0.17	0.08

5.2 Evaluation

5.2.1 Quantitative

The overall results for the 'human judge' test was 0.25 for precision and 0.24 for recall. This means that there was roughly a 25% correspondence between both the number of key terms generated by the system also appearing in the manually compiled list of terms in the title and abstract, and also with the number of key term clusters in the compiled list which appeared in the system's list. In order to place these results in context, lower and upper bounds were determined (test results in Appendix G) for both precision and recall. To determine the lower bound, 10 randomly chosen Noun Phrases were selected in each of the 15 documents and in each case, were compared with the terms generated by the system. The lower bound for precision and recall were both calculated as 1.3%. The upper bounds for precision and recall were established by comparing the two human judges' lists. Although comparing only two human judges is by no means conclusive, the results give an indicator to the highest achievable accuracy of the system's term

extraction method. The upper bounds serve as a benchmark from which to compare the results of the 'human judge' test. After comparing the two human judges lists, the upper bounds for precision and recall were calculated as 53% and 51% respectively.

The results of the 'Human Judge' test show that the accuracy of the system lies roughly half way between the lower and upper bounds. The system easily outperforms the random, lower bound test, being some 23-24% more accurate in both precision and recall. Against the upper bound, there is again a discrepancy, with the system performing around 27-28% lower for both precision and recall. Several limitations of the system as discussed in detail in the subsequent section (5.2.2 Qualitative), offer explanations as to why both measures showed moderate accuracy.

The task of a human judge extracting 10 document defining key terms is a lengthy process taking between 1 – 2 hours to complete. For this reason, the time scale for this phase of the project permitted only two human judges to take part. The results must therefore not be taken as the definitive measure for determining the accuracy of the implemented term extraction method. The subjective nature of this test means that the two human judges may not be representative of the overall norm i.e. the standardised set of key terms that would be extracted by the majority of conference presenters.

Human judges in identifying key terms of a given scientific article, tend to pick out mostly abbreviations and technical terms. As covered in the follow section (5.2.2. Qualitative), this system cannot handle such terminology during the tagging stage, due to the lack of domain specific knowledge (science and NLP) in both the CorpusTagTr and in the vocabulary of Wordnet (the dictionary used in the tagging process). The scope of the system is therefore somewhat limited to mostly terms, which comprise of commonly used English words.

The way in which a human judge goes about finding key terms as opposed to the system is subtly different. Human judges pick out some terms which occur very few times in the document, but which encapsulate the semantics of the content. The TF.IDF function gives a higher weighting to those terms appearing frequently within the source document. Although the system may have extracted many of the key terms that the user identified, this discrepancy in weighting means that setting the cut-off point for the number of terms to be returned to ten, may result in a low precision score. The results do not give the complete picture as to the overall recall since in reality, the system can return more terms, dependent on what the user selects for the '% of key terms returned' variable.

The results for the second test, 'Title and Abstract' show a slightly lower accuracy than that of the 'Human Judge' test. The precision of the system on this test, was found to be 0.17 while the recall was only 0.08. Obtaining lower scores for the measures than in the previous test was to be expected. This test only focuses on the title and abstract of a document whereas the 'Human Judge' test allows terms to be sourced from both the abstract and main body of the document. Since the system can also identify terms

from both the abstract and main body, it is likely that many terms generated may derive purely from the main body and do not feature in the title or abstract.

When an author writes an abstract, their aim is to summarise into a small number of sentences (usually between 3-10), the purpose of the document. In general, the title and abstract uses concise, technical terminology, which may not heavily feature in the main text. On the assumption that the system was able to pick up the domain specific terminology encapsulated in the title and abstract, since they will likely occur infrequently in the document as a whole, the system gives such terms a low (TF.IDF) ranking and this may partially explain why some terminology is not included in the top ten returned by the system.

In addition, the TF.IDF measure only looks at terms appearing in at least two documents (the source document and at least one other in the CorpusSciTR). Since the abstract aims to use terminology to uniquely characterise a document, many important terms are excluded in the TF.IDF comparison since they only appear in the source document. The system ominously ignores such terms even though in actual fact they are strong candidate terms for characterising the semantics of that document.

5.2.2 Qualitative

Identifying the types of errors/inaccuracies and where they stem from in the term extraction method will be analysed in detail during this section. The errors and limitations that exist in this method can be classified into the categories of ‘tagging errors’, ‘Wordnet limitations’, ‘chunking limitations’ and ‘the TF.IDF function’. Tagging errors concerns identifying any inaccuracies that exist as a result of the Unigram and Brill taggers either failing to or wrongly tagging words. Wordnet limitations concern the restrictive vocabulary of the dictionary that is unable to identify and tag some of the technical scientific words. Limitations of the chunking function look at the formulation of finding key terms and also the lack of sophistication to be able to apply inference when returning noun phrases. The final classification, ‘the TF.IDF function’, briefly highlights its restriction as previously mentioned in the quantitative section (5.2.1).

Although the accuracy previously determined for the Brill tagger is high (around 0.95 or 95%), there still remains numerous instances where the tagger has either failed to tag or wrongly tagged words. The major limitation of the taggers was the domain of the corpora on which it was trained. As no specific scientific corpora formatted in the style to be used by the NLTK toolkit exists, a number of large corpora encapsulating the generic domain of the (American-)English language had to suffice. This meant accepting that the Brill tagger would not be able to tag scientific words embodied in the CorpusSci, but not existing in the CorpusTagTr, at least not before transformation rules have been applied. After identifying and applying a number of transformation rules, a proportion of these previously untagged words were given tags. Where a sequence of untagged scientific words existed e.g. ‘class-based n-gram’, the transformation rules had little or no impact.

The use of the transformation rules as used by the Brill tagger not only assisted in tagging some previously untagged words, but equally as crucial, re-assigned the tags of words having ambiguous POS e.g. applying a transformation rule to alter the tagging of 'state' in 'The authors state…', from 'NN' to 'VB'. Although most reassignments were correctly altered, a small quantity of words having either previously been tagged or up to that point, had no tag, were wrongly tagged. One such error was found with the word 'apply' which can be either a 'VB' (verb base) or 'VBP' (verb, non-3rd person, present tense). In the phrase 'we apply estimation methods', the Brill tagger assigned the 'VB' POS tag. The reassignment should have been to 'VBP'.

The proportion of tags identified by the Unigram and Brill taggers could have been reduced marginally if a larger CorpusTagTr had been used. The task of training the Brill tagger, although performed one off, was a slow operation. This greatly restricted the size of the CorpusTagTr. Even without such a restriction, as the CorpusTagTr characterises the English language without focussing on any specific domains or fields, using a larger corpora of the same or similar domain would have little added value in terms of increasing the number of tagged words. Extending the CorpusTagTr with documents focussing on the scientific domains/fields covered by CorpusSci on the other hand would have had a profound effect, being able to tag a greater proportion of the technical content.

After attempting to tag words using the taggers, a few post-processing operations were performed to tag as many untagged words as possible. Wordnet (an electronic dictionary) was used in all these operations to look up and find the POS of words. As was true of the CorpusTagTr, the domain knowledge of Wordnet does not include many niche, scientific terms used in the CorpusSci. For this reason, the effect of utilising this dictionary was minimal. In addition, scientific articles contain many abbreviations. Wordnet does not support abbreviations. Even if support was included, as many of the terms abbreviated correspond to technical phrases, it is likely that in their expanded form, a large proportion of these words would be outside of Wordnet's knowledge base.

Another limitation arises from authors using hyphens in hyphenated words subjectively. An example found in one particular document is the word 'covariation'. Wordnet has no knowledge of hyphenated words, which have been merged into a single word i.e. no hyphen used. If the author in question had instead written 'co-variation', a regular expression would have been able to identify the word proceeding the hyphen and looked it up in Wordnet. Since 'variation' is in Wordnet's vocabulary, the complete hyphenated word would be tagged as a 'NN' i.e. a noun. Attempting to write logic (regular expressions) to add hyphens to words that the author has written as a single word would be a complex operation beyond the scope of this project and would in any event make little difference to reducing the number of untagged words.

The next operation in the term extraction system having a number of associated limitations was the chunking function. The Noun Phrase chucker looks for very specific words and phrases. In general, there are other types of phrases, which may be of interest when returning key terms to be used within a conference

presentation. A significant limitation of the chunker is the fact that prepositions are ignored when identifying phrases. There are countless examples in the CorpusSci where two noun phrases are for example joined by the preposition 'of' e.g. 'the problem *of* interfacing morphology'. The NP chunker used in this system would class such terms as two separate phrases i.e. 'the problem' and 'interfacing morphology', instead of simply as a single phrase.

With conjunctions, the chunker is not sophisticated to the point of being able to return two separate noun phrases, both containing the same associated noun, when the conjunction is in the form of 'adjective(x) and adjective(y) noun'. Presently the chunker returns two phrases, 'x' and separately 'y noun'. One such example is in the phrase 'database and input expressions', the chunker returns the phrases 'database' and 'input expressions'. No context is given to the first adjective. The improvement would be to return two phrases of the form 'adjective noun' i.e. 'database expressions' and 'input expressions.

The remaining function of the term extraction method that is restrictive, stems from using the TF.IDF algorithm. As the algorithm requires at least one other document in the CorpusSciTR to contain a given word, as well as being present in the current source document, a significant proportion of terms only occurring in the source document are ignored. Since the aim of this method is to generate a list of terms uniquely characterising a particular document, a proportion of candidate terms are unfairly discarded. This is a weakness of this system and future improvements should seek to combine the TF.IDF method with other measures allowing for terms appearing only in the source document to be included when identifying the most relevant key terms for a given document.

Section 6: Conclusions

6.1 Achievement of Project Aims

6.1.1 Minimum Requirements

In completing this project, all the minimum requirements (as specified in section 1.3) necessary to research, implement, test and evaluate the main functionality supported by this system have been fulfilled.

The first phase of this project involved researching current IR and terminology extraction techniques and determining their applicability for the purposes of developing a system to semi automate the generation of slideshows. Section 2 (Background Reading) of this report outlines in detail the relevant topics/methods in the fields of IR and term extraction, ending with a justification into the adoption of specific methods for this project. The contents of this chapter specifically fulfil the first identified minimum requirement.

Having chosen certain NLP techniques/methods and after deciding on a software methodology, programming languages and a framework/guidelines for the UI, the first phase within the implementation section was to develop a term extraction method. As specified in the second requirement, the term extraction method is able to identify words and terms. The requirement also specifies formula, but since the DTD scheme (Appendix C) used in the CorpusSci contains a tag (as opposed to the actual formulae) to indicate the placement of equations/formulae, no such information can be extracted. All identified terms are filtered to return a subset of 'key terms' based upon a cut-off point identified through the use of the TF.IDF method. As previously stated in the qualitative evaluation section (5.2.2), the system is not without restrictions when identifying key terms as only those terms which appear in both the source document and at least one other document in the CorpusSciTR, are included.

The User Interface was then implemented, following the framework/guidelines specified in the User Interface Design of the Design section (3.4). This supported the underlying functionality of the term extraction method as well as the editing features and 'flexibility' options. Options for flexibility included allowing the user to change the quantity of key-terms to be returned (a slider bar using % unit of measurement) and the number of further research materials to be returned relating to the contents of the source document. The third minimum requirement additionally specified that the user should be offered the flexibility to be able alter the length of key-terms to be returned. This feature was not included in the finished system since during intermediary testing, the length of key terms typically returned ranged between 1-3. Allowing the user to restrict the length with such a small range would be of little use as most terms are single words and those that are two or more are mainly document defining key terms.

As early as the background stage, finding a corpus suitable for use in this project became a crucial task (identified as the fourth minimum requirement). Although domain independent techniques have been adopted as best as possible, the structure and layout of documents was a factor in determining which corpus to adopt. The writer by chance found an appropriate corpus early on thanks to the help of his supervisor. An XML corpus was located containing documents all within the domain of science (mostly more specifically within the field of NLP). The additional information afforded by the XML e.g. subheadings, title and author was beneficial to both the segmentation method as well as presenting the user an introduction slide at the beginning of every generated presentation.

After finding a suitable corpus (CorpusSci), the documents had to be divided into two parts. A sub selection was used as training material on which Part Of Speech tagging was performed to provide the system vocabulary related to the domain of source documents. The remaining documents of the CorpusSci were used for both intermediary testing and then at the end of the implementation section, for performing the 'title and abstract' and 'human judge' test. The final minimum requirement was fulfilled within the previous chapter i.e. evaluation (section 6.2). This provided both an evaluation based directly from the interpretation of the results of the two tests performed as well as a qualitative analysis into the limitations and categories of errors that exist in the functionality of the term extraction method.

6.1.2 Extensions

Out of the three suggested extensions (section 1.3), only the second extension of presenting the generated content in slide format was implemented. Although through the background reading section, 'TextTiling' was found to be an appropriate technique for use in segmenting the key terms into slides, at the point of implementation when this method was to be implemented, time restrictions meant that a slightly simpler option had to be developed in order to meet project deadlines. The method used as detailed in section 4.4 (Slide Formatting), is better known as sectionalising i.e. the process of categorising instances of key terms into slides based upon the subheading they appear under. Each subheading is given a slide with all its associated key terms. This method was supported in the UI by providing navigational controls (buttons) to allow the user to change the current slide being viewed in the main text area as well as supporting features to allow the user to add/remove slides and export the presentation contents to Open Office's presentation application (Impress).

Although key terms are useful in constructing presentations, of equal benefit is identifying key sentences of an article. Providing the user the flexibility to view key terms or key sentences or a combination of both would have been a better solution. This project, having a large scope for development meant that the extension of sentence extraction (summarisation) could not be implemented in the time scale given.

The other suggested extension of accepting plain text (non marked-up) domain related documents and converting them into marked-up XML documents also did not get implemented. The problem of being able to add certain tagging information such as sub-headings and titles would be extremely difficult

unless special characters or a specific layout e.g. two lines spacing between each subheading and its first paragraph; was conformed with. This may be a minor area for future development though it was not considered to be of real importance to meeting the overall aims of this project.

A small extension not suggested as a possible extension was added towards the end of the implementation phase. As well as generating a series of slides for a presentation, the implemented extension provided URLs pointing to further research material relevant to a given source document.

A further extension also added towards the end of implementation was the facility to export the currently opened presentation into Open Office.org's Presentation Application (Impress). This required creating a number of XML files, zipping them into a single file and converting the file extension. Once read into Impress, presentations can then be converted into Microsoft's PowerPoint format i.e. '.ppt'.

6.2 Scope for other domains

As mentioned throughout the report, the aim in designing and implementing the system was to as far as possible, use domain independent techniques. This decision was made with the thought that future work/projects might take place to adapt this system to work with other domains.

In order to use this system with source documents from a different domain, a number of alterations need to be made to the code used within the system. The hierarchical based structure by which the files interact with each other means that all programming alterations required to change the domain of interest are confined to a single file.

The crucial part for adding support for other domains is the adoption of an appropriate corpus. A programmer will need to find a corpus, which reflects the domain(s) of the source document(s). Equally as fundamental, unless support is added for converting plain text files (with basic formatting) into XML documents conforming to the DTD scheme used in CorpusSci (Appendix D), any corpus chosen by the programmer must be formatted to the XML format used in CorpusSci.

The other changes concern altering a couple of variables in a single file (lectureterms.py), to assist in the tagging process. Although the main source for tagging any domain specific corpus comes from the corpus used to train the tagger(s) i.e. CorpusTagTr, as well as from Wordnet's vocabulary, there may be a number of important terms to the domain being adopted that are still outside of the system's knowledge. For this reason, the programmer can manually add as many domain terms and their associated POS tags as is necessary. Any terms manually added by the programmer should be done with due care to ensure words or phrases with ambiguous POS are not included.

As opposed to just altering the domain adopted to a different one, the system could ideally be extended to include corpora from many domains. A user through the UI, could select the domain of the

source document and the system would compare all identified terms in the source document with the terms belonging to the corresponding domain related corpus/corpora.

6.3 Future Work

With regard to directions by which future development of this project could take, there are many possibilities, so much so that it would be difficult to list them all, let alone know them all. Scope for future projects, which the writer has identified, will be discussed during this section of the report.

One possible direction to extend this project would be to combine the collocations method with the current term extraction method. This will allow the system to be able to identify commonly co-occurring n-grams, which are not necessarily noun phrases. This would assist in such problems as identified in the qualitative section of the evaluation where for example many phrases of interest, which were discarded by the system, contained prepositions such as 'of'. The other facet of collocations is that n-grams whose words do not all appear adjacent to each other can be identified. How the two techniques of collocations and term extraction would be combined is a major consideration in itself.

Another direction for extending this project regards the weighting scheme used for identifying terms as being key terms of a source document. Currently, an arbitrary weighting based upon intermediary testing involving looking at the terms generated by system, is used. This weighting gives terms consisting of two words or more a factor of three times greater importance than single word terms. This is obviously static and does not adapt to different documents. A better solution would be to develop some algorithm, which calculated the weighting of multiple word terms against single word terms, based upon the contents of the document e.g. determining the percentage of identified terms (Noun Phrases) that are greater than one word.

Using or combining a different weighting scheme could overcome the limitation of the current weighting scheme identified in the qualitative section of the evaluation. The TF.IDF scheme ignores terms occurring only in the source document, but not in any other document in the CorpusSciTR. When analysing alternative weighting schemes, need to bear in mind that any adopted scheme will have to be adapted to find 'key terms' as opposed to as is usual in IR, 'documents'.

The electronic dictionary Wordnet is currently used for just finding the POS of those words, which failed to be tagged by the Brill Tagger. The dictionary supports numerous other lookup features, which could be exploited in future development. Having extracted a list of key terms from both the source document and CorpusSciTR, individual terms could be looked up through Wordnet to find for example, synonyms and hypernyms. The synonyms of all key terms identified in the source document, could be included in the source document's term frequency index. This expanded index would assist in improving the accuracy of the term extraction system, increasing the number of possible terms from which to filter using some weighting scheme (TF.IDF). Hypernyms could be used to try to establish links between key terms identified in the source document and in CorpusSciTR. Hypernyms are words which encapsulate the meaning of a number of

other words e.g. ‘person’ is a hypernym of both ‘man’ and ‘woman. For each key term in the source document, its hypernym could be looked up and compared with all the hypernyms of key terms in CorpusSciTR. Any pair of key terms with a matching superordinate may be included in the TF.IDF algorithm for comparison. There are many possibilities as to the exact nature by which to implement a weighting scheme reflecting the use of hypernyms. Consideration should be given to the number of layers, which link a pair of words with their common superordinate.

Both suggested extensions, which were not implemented during this project, would serve as valuable additions for improving the quality of the generated slideshow content and increasing the flexibility of document formats, which could be read into the system. As mentioned in section 6.1.2 (extensions), adopting sentence extraction in conjunction with term extraction and allowing the user to control the quantity (%) of each to be returned, would better serve in reducing the amount of editing required to construct the slideshow from this raw form as well as minimising the semantics of the source document that are lost. The other suggested extension i.e. converting raw-texts into marked-up XML documents would be a minor point of improvement. The plain texts being converted would have to have some standardised formatting to allow regular expressions to mark-up (XML) such information as the title, subheadings and start and end of paragraphs. In converting the plain text to XML format, only the mandatory tags need be added. Other optional tags should be added at the programmer’s discretion as long as the DTD scheme as used in the CorpusSci is adhered to.

In implementing the extension to segment the list of identified key terms, a simplified method known as sectionalising was adopted instead of the ‘TextTiling’ method discussed in the background reading section (2.4). Future development could seek to combine ‘sectionalising’ with the TextTiling method. Sectionalising could first subdivide the terms into slides with associated headings. The TextTiling technique could then identify those slides containing more terms than some determined threshold and subdivide such slides into a number of slides, based upon the number of topic boundaries discovered.

An entirely different direction for future projects may be to allow users to add their own domains by which the system can handle. In the previous section (6.2), support for adding different domains hinged around the programmer hard coding some alterations. An improvement to the system could be made to allow the user to add support for new domains of their choice. Providing users have a corpus in the specific XML format or in the plain text format, which can be converted to an XML format, then through the UI and its associated underlying functionality, support could be added to allow the user to add a new domain that is not currently in the system’s knowledge repository. Once a user has added several new domains, subsequent executions of the program would allow a user to use source documents that are in any of the supported domains.

A final scope for development would be to use an XML parser for the purposes of extracting specific information e.g. main content, title and subheadings. This operation is currently achieved through the use of many regular expressions. An XML parser is more future proof (flexible) if changes are made to the DTD

scheme. Regular expressions can be considered both static and rigid, always looking for specific patterns. Using an XML parser will allow for example, a user to add a new variable to a tag e.g. ‘number’ in ‘<header number=1>’, while still without making any alterations, be able to parse the contents embedded within such tags.

Bibliography

McEnery, T. & Wilson, A. (2001). *Corpus Linguistics*. Edinburgh: Edinburgh University Press.

Teufel, S. (1999). *Argumentative Zoning: Information Extraction from Scientific Text*. [online]. Available from: http://www.cl.cam.ac.uk/users/sht25/thesis/t.pdf [Accessed: 13th April 2005].

Manning, C.D. & Schutze, H. (2000). *Foundations of Statistical Natural Language Processing* (2nd Printing). Cambridge (Massachusetts, U.S.): The MIT Press.

Smadja, F. (1993). *Retrieving Collocations from Text: Xtract*. In Computational Linguistics (vol.19, num. 1, March 1993).

Daille, B. (1994). ACL Anthology. *Study and Implementation of Combined Techniques for Automatic Extraction of Terminology*. In The Balancing Act: Combining Symbolic and Statistical Approaches to Language.

Ananiadou, S. (1994). ACL Anthology. *A Methodology for Automatic Term Recognition*. In 15th International Conference on Computational Linguistics (COLING, vol. 2, 1994).

Jacquemin, C. (1994). LIMSI. *Term Extraction and Automatic Indexing*. [online]. Available from: http://www.limsi.fr/Individu/jacquemi/FTP/JacBourHandbookCL.pdf [Accessed: 21st November 2004].

Hearst, M. A. (1997). *TextTiling: Segmenting Text into multi-paragraph subtopic passages*. In Computational Linguistics (vol. 23, num. 1, March 1997).

What is Jython? (2003), (Jython.org), Available from: http://www.jython.org/docs/whatis.html [Accessed: 10th April 2005].

Avison, D. E. & Fitzgerald, G. (1995). *INFORMATION SYSTEMS DEVELOPMENT: Methodologies, Techniques and Tools*. London: McGraw-Hill.

Avgeron, C. & Cornford, T. (1998). *DEVELOPING INFORMATION SYSTEMS – Concepts, Issues and Practice,* 2nd ed, London: MacMillan Press Ltd.

Raccoon, L. B. S. (1996). *Fifty Years of Progress in Software Engineering*. In ACM SIGSOFT Software Engineering Notes (vol.22, issue 1, January 1997).

Boehm, B. W. (1988). *A Spiral Model of Software Development and Enhancement*. IEEE Computer (vol. 21, num. 5, p. 61-72, May 1988).

Shneiderman, B. & Plaisant, C. (2005). *Designing The User Interface: Strategies for Effective Human-Computer Interaction,* 4th ed, Boston: Addison Wesley.

Nielsen, J. (1994). *Enhancing the explanatory power of usability heuristics*. In Proc. ACM CHI'94 Conf. (p. 152-158, April 1994).

Francis, W. N. & Kucera, H. (1979). ICAME. *Brown Corpus Manual*. [online]. Available from: http://helmer.aksis.uib.no/icame/brown/bcm.html [Accessed: 30th March 2005].

Barrett, A. D. (2004), *Lego Mindstorms Robotic Football - Final Year Project*. School of Computing, University of Leeds, Appendix A.

Appendix A – Personal Reflection

Advice for Students Undertaking Similar Projects

For any project involving a lengthy implementation phase, requiring the adoption of complex algorithms and techniques with which the developer is not familiar, it is important to avoid rigidly following a life cycle structure, especially ones which promote sequential transition between phases. Although it is important to have a good understanding of concepts and methods to be utilised in the implementation, starting this phase as soon as possible is essential. The use of a new toolkit, which had not been fully debugged, had little documentation and a small user community, required longer than expected man-hours to understand its functionality. Starting the implementation at an early stage also assisted in improving my understanding of some of the more trickier theoretical concepts of NLP. The reality of developing a deeper understanding in a particular domain is an iterative process between researching domain related literature and applying the learnt techniques in the implementation phase. Through understanding the theory behind the techniques used during implementation, a developer can then further their knowledge through successive iterations of literature search and implementation.

During the course of implementing the system, there were a number of occasions where communication via e-mail to fellow adopters (and developers) of the NLTK toolkit was required. Initially I had difficulty in obtaining any response. From the correspondence that took place, I quickly understood that the way in which I structured the content of emails, had a profound effect on both the rate and quality of responses. When seeking technical advice, it is important to give the recipient some background to the FYP and state for what purposes the toolkits/libraries/languages are intended. Developers and users are usually students or professionals with limited time and it is therefore important to portray a professional attitude, letting them know you are a student and are making use of the tools/languages as part of your FYP. The recipient should also be provided with a detailed but concise background to the problems for which you are seeking their advice.

General Advice for All Future Students

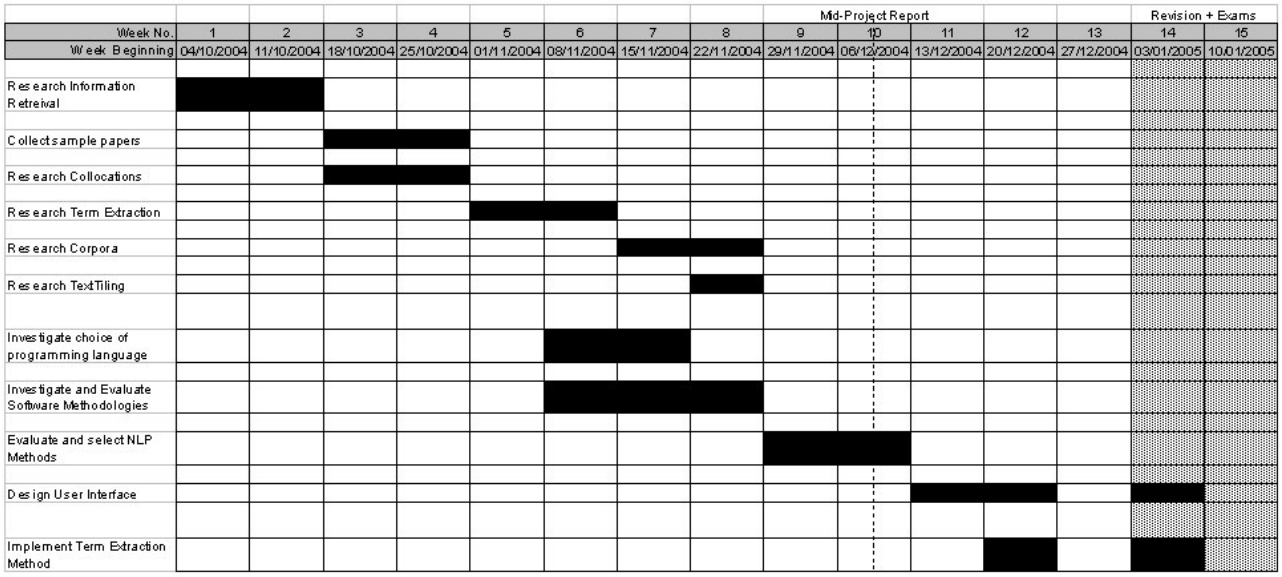
As pointed out by Barrett (2004), the help afforded through supervisors is invaluable and future students should seek to make full use of their knowledge and feedback. Dependent on the supervisor you are assigned, will affect the type of assistance on offer. In all cases, supervisors should be your first reference regarding where to search for appropriate literature, what should go in your report and the precise nature by which you should plan and structure the report.

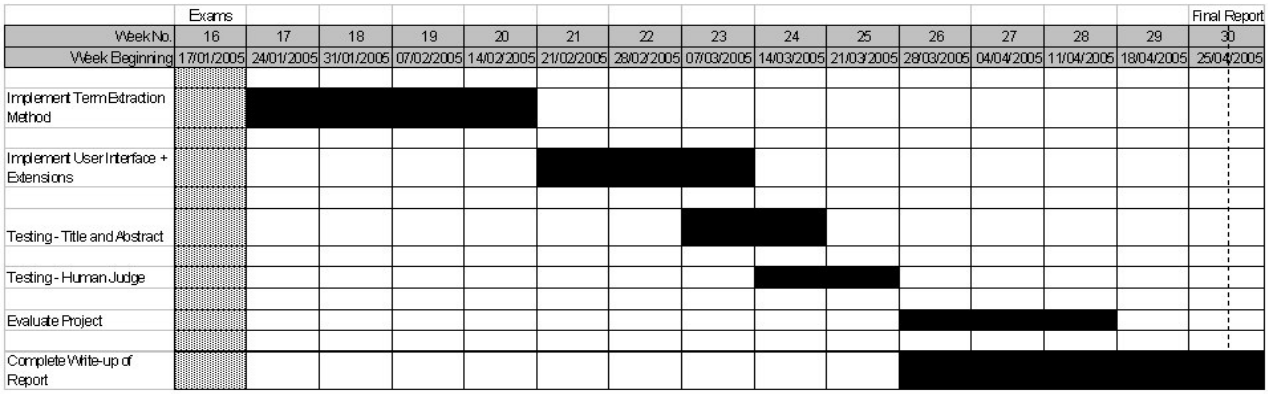
In many cases, the supervisor will possess specific knowledge to the domain of your FYP. In such cases, the help received is two-fold since they can also assist in your understanding of the problem and provide critical feedback for any parts of your report, which you should submit as draft chapters.

Weekly meetings with the supervisor were of great importance in clarifying any problems encountered as well as demonstrating progress. Before going to meetings, future students should have in mind exactly what they want to discuss or review. Meetings are not meant for students to just sit back and listen to their supervisor, but rather are for students to discuss their concerns and ideas, to be able to achieve any goals set for the forthcoming week.

One of the first tasks that should be carried out during background reading is to develop a rough structure for your FYP report with the TOC in mind. Once completed, an initial project schedule should be constructed. It is important to complete these two tasks as early as possible, in order that weekly goals can then be set with regard to what tasks you realistically expect to complete. The project schedule should be constructed taking into account any disparity between the number of modules studied in each of the two semesters. In this project, I had fewer modules in the second semester and as can be gleaned from the project schedule (Appendix B), the workload, in particular the implementation, was heavily weighted towards the latter semester.

Appendix B – Project Schedule (Gantt chart)





Appendix C – DTD Scheme for Scientific Articles Corpus

```
<!ELEMENT PAPER (TITLE,REFLABEL,AUTHORS,FILENO,APPEARED,ANNOTATOR?,DATE?,ABSTRACT,
BODY,REFERENCES?)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT AUTHORS (AUTHOR+)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ELEMENT FILENO (#PCDATA)>
<!ELEMENT ANNOTATOR (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT YEAR (#PCDATA)>
<!ELEMENT APPEARED (#PCDATA)>
<!ELEMENT EQN EMPTY>
<!ATTLIST EQN
C CDATA 'NP'>
<!ELEMENT CREF EMPTY>
<!ATTLIST CREF
C CDATA 'NP'>
<!ELEMENT REFERENCES (P|REFERENCE)*>
<!ELEMENT REFERENCE (#PCDATA|REFLABEL|W|EQN|NAME|SURNAME|DATE|ETAL|REFAUTHOR|YEAR)*>
<!ELEMENT NAME (#PCDATA|SURNAME|INVERTED)* >
<!ELEMENT SURNAME (#PCDATA)>
<!ELEMENT REF (#PCDATA)*>
<!ATTLIST REF
SELF (YES|NO) "NO"
C CDATA 'NNP'>
<!ELEMENT REFAUTHOR (#PCDATA|SURNAME)*>
<!ATTLIST REFAUTHOR
C CDATA 'NNP'>
<!ELEMENT ETAL (#PCDATA)>
<!ELEMENT BODY (DIV)+>
<!ELEMENT DIV (HEADER?, (DIV|P|IMAGE|EXAMPLE)*)>
<!ATTLIST DIV
DEPTH CDATA #REQUIRED >
<!ELEMENT HEADER (#PCDATA|EQN|REF|REFAUTHOR|CREF|W)*>
<!ATTLIST HEADER ID ID #REQUIRED >
<!ELEMENT P (S|IMAGE|EXAMPLE)*>
<!ATTLIST P
TYPE (ITEM|TXT) "TXT">
<!ELEMENT IMAGE EMPTY>
<!ATTLIST IMAGE
ID ID #REQUIRED
CATEGORY (AIM|CONTRAST|TEXTUAL|OWN|BACKGROUND|BASIS|OTHER) #IMPLIED>
<!ELEMENT S (#PCDATA|EQN|REF|REFAUTHOR|CREF|FORMULAIC|AGENT|FINITE|W)*>
<!ATTLIST S
TYPE (ITEM|TXT) "TXT"
ID ID #REQUIRED
ABSTRACTC CDATA #IMPLIED
CATEGORY (AIM|CONTRAST|TEXTUAL|OWN|BACKGROUND|BASIS|OTHER) #IMPLIED>
<!ELEMENT ABSTRACT (A-S)*>
<!ELEMENT A-S (#PCDATA|EQN|REF|REFAUTHOR|CREF|FORMULAIC|AGENT|FINITE|W)*>
<!ATTLIST A-S
ID ID #REQUIRED
TYPE (ITEM|TXT) "TXT"
DOCUMENTC CDATA #IMPLIED
CATEGORY (AIM|CONTRAST|TEXTUAL|OWN|BACKGROUND|BASIS|OTHER) #IMPLIED>
<!ELEMENT EXAMPLE (EX-S)+>
<!ATTLIST EXAMPLE
ID ID #REQUIRED
CATEGORY (AIM|CONTRAST|TEXTUAL|OWN|BACKGROUND|BASIS|OTHER) #IMPLIED>
<!ELEMENT EX-S (#PCDATA|EQN|W)*>
<!ELEMENT W (#PCDATA)>
<!ATTLIST W
C CDATA #IMPLIED>
<!ELEMENT FINITE_VERB (#PCDATA)>
```

```
<!ATTLIST FINITE_VERB
ACTION
(AFFECT_ACTION|ARGUMENTATION_ACTION|AWARE_ACTION|BETTER_SOLUTION_ACTION|CHANGE_ACTION|
COMPARISON_ACTION|CONTINUE_ACTION|CONTRAST_ACTION|FUTURE_INTEREST_ACTION|INTEREST_ACTION|
NEED_ACTION|PRESENTATION_ACTION|PROBLEM_ACTION|RESEARCH_ACTION|SIMILAR_ACTION|
SOLUTION_ACTION|TEXTSTRUCTURE_ACTION|USE_ACTION|POSSESSION|COPULA|0)
"0">
<!ELEMENT FORMULAIC (#PCDATA|EQN|CREF|REF|REFAUTHOR)*>
<!ATTLIST FORMULAIC TYPE
(US_AGENT|REF_US_AGENT|REF_AGENT|OUR_AIM_AGENT|US_PREVIOUS_AGENT|THEM_PRONOUN_AGENT|THEM_
AGENT|
GENERAL_AGENT|PROBLEM_AGENT|SOLUTION_AGENT|THEM_FORMULAIC|US_PREVIOUS_FORMULAIC|
TEXTSTRUCTURE_AGENT|NO_TEXTSTRUCTURE_FORMULAIC|IN_ORDER_TO_FORMULAIC|AIM_FORMULAIC|
TEXTSTRUCTURE_FORMULAIC|METHOD_FORMULAIC|HERE_FORMULAIC|CONTINUE_FORMULAIC|SIMILARITY_FOR
MULAIC|
COMPARISON_FORMULAIC|CONTRAST_FORMULAIC|GAP_FORMULAIC|FUTURE_FORMULAIC|AFFECT_FORMULAIC|
GOOD_FORMULAIC|BAD_FORMULAIC|0)
"0">
<!ELEMENT AGENT (#PCDATA|EQN|REF|CREF|REFAUTHOR)*>
<!ATTLIST AGENT
TYPE
(US_AGENT|THEM_AGENT|THEM_PRONOUN_AGENT|US_PREVIOUS_AGENT|REF_US_AGENT|REF_AGENT|
GENERAL_AGENT|PROBLEM_AGENT|SOLUTION_AGENT|0) "0">
```

Appendix D – Sample XML Document from Scientific Articles Corpus

```
<?xml version='1.0'?>
<!DOCTYPE PAPER SYSTEM "/proj/nlp/users/teufel/dtd/s.dtd" >
<PAPER>
<FILENO>9505037</FILENO>
<TITLE>Identifying Word Translations in Non-Parallel Texts</TITLE>
<AUTHORS>
<AUTHOR>Reinhard Rapp</AUTHOR>
</AUTHORS>
<APPEARED>ACL95</APPEARED>
<CLASSIFICATION> Lg.Pr.Cl </CLASSIFICATION>
<ABSTRACT>

<A-S ID='A-0'> Common algorithms for sentence and word-alignment allow the automatic
identification of word translations from parallel texts .</A-S>
<A-S ID='A-1'> This study suggests that the identification of word translations should
also be possible with non-parallel and even unrelated texts .</A-S>
<A-S ID='A-2'> The method proposed is based on the assumption that there is a correlation
between the patterns of word co-occurrences in texts of different languages .</A-S>
</ABSTRACT>
<BODY>
<DIV DEPTH='1' >
<HEADER ID='H-0'> Introduction </HEADER>
<P>
<S ID='S-0'> In a number of recent studies it has been shown that word translations can
be automatically derived from the statistical distribution of words in bilingual parallel
texts (e. g. <REF>Catizone et al. 1989</REF> <REF>Brown et al. 1990</REF>; <REF>Dagan et
al. 1993</REF><REF>Kay and Roescheisen, 1993</REF> ) .</S>
<S ID='S-1'> Most of the proposed algorithms first conduct an alignment of sentences, i.
e. those pairs of sentences are located that are translations of each other .</S>
<S ID='S-2'> In a second step a word alignment is performed by analyzing the
correspondences of words in each pair of sentences .</S>
</P>
<P>
<S ID='S-3'> The results achieved with these algorithms have been found useful for the
compilation of dictionaries, for checking the consistency of terminological usage in
translations, and for assisting the terminological work of translators and interpreters
.</S>
</P>
<P>
<S ID='S-4'> However, despite serious efforts in the compilation of corpora (<REF>Church
and Mercer 1993</REF>;<REF>Armstrong and Thompson 1995</REF>) the availability of a large
enough parallel corpus in a specific field and for a given pair of languages will always
be the exception, not the rule .</S>
<S ID='S-5'> Since the acquisition of non-parallel texts is usually much easier, it would
be desirable to have a program that can determine the translations of words from
comparable or even unrelated texts .</S>
</P>
</DIV>
<DIV DEPTH='1' >
<HEADER ID='H-1'> Approach </HEADER>
<P>
<S ID='S-6'> It is assumed that there is a correlation between the co-occurrences of
words which are translations of each other .</S>
<S ID='S-7'> If - for example - in a text of one language two words A and B co-occur more
often than expected from chance, then in a text of another language those words which are
translations of <REFAUTHOR>A</REFAUTHOR> and B should also co-occur more frequently than
expected .</S>
<S ID='S-8'> This assumption is reasonable for parallel texts .</S>
<S ID='S-9'> However, in this paper it is further assumed that the co-occurrence patterns
in original texts are not fundamentally different from those in translated texts .</S>
</P>
<P>
```

<S ID='S-10'> Starting from an English vocabulary of six words and the corresponding German translations, table <CREF/> show an English and a German co-occurrence matrix .</S>
<S ID='S-11'> In these matrices the entries belonging to those pairs of words that in texts co-occur more frequently than expected have been marked with a dot .</S>
<S ID='S-12'> In general, word order in the lines and columns of a co-occurrence matrix is independent of each other, but for the purpose of this paper can always be assumed to be equal without loss of generality .</S>
</P>
<P>
<S ID='S-13'> If now the word order of the English matrix is permuted until the resulting pattern of dots is most similar to that of the German matrix (see table <CREF/>), then this increases the likelihood that the English and German words are in corresponding order .</S>
<S ID='S-14'> Word n in the English matrix is then the translation of word n in the German matrix .</S>
</P>
</DIV>
<DIV DEPTH='1' >
<HEADER ID='H-2'> Simulation </HEADER>
<P>
<S ID='S-15'> <REFAUTHOR>A</REFAUTHOR> simulation experiment was conducted in order to see whether the above assumptions concerning the similarity of co-occurrence patterns actually hold .</S>
<S ID='S-16'> In this experiment, for an equivalent English and German vocabulary two co-occurrence matrices were computed and then compared .</S>
<S ID='S-17'> As the English vocabulary a list of 100 words was used, which had been suggested by <REF>Kent and Rosanoff 1910</REF> for association experiments .</S>
<S ID='S-18'> The German vocabulary consisted of one by one translations of these words as chosen by <REF>Russell 1970</REF> .</S>

</P>
<P>
<S ID='S-19'> The word co-occurrences were computed on the basis of an English corpus of 33 and a German corpus of 46 million words .</S>
<S ID='S-20'> The English corpus consists of the Brown Corpus, texts from the Wall Street Journal, Grolier's Electronic Encyclopedia and scientific abstracts from different fields .</S>
<S ID='S-21'> The German corpus is a compilation of mainly newspaper texts from Frankfurter Rundschau, Die Zeit and Mannheimer Morgen .</S>
<S ID='S-22'> To the knowledge of the author, the English and German corpora contain no parallel passages .</S>
</P>
<P>
<S ID='S-23'> For each pair of words in the English vocabulary its frequency of common occurrence in the English corpus was counted .</S>
<S ID='S-24'> The common occurrence of two words was defined as both words being separated by at most 11 other words .</S>
<S ID='S-25'> The co-occurrence frequencies obtained in this way were used to build up the English matrix .</S>
<S ID='S-26'> Equivalently, the German co-occurrence matrix was created by counting the co-occurrences of German word pairs in the German corpus .</S>
<S ID='S-27'> As a starting point, word order in the two matrices was chosen such that word n in the German matrix was the translation of word n in the English matrix .</S>
</P>
<P>
<S ID='S-28'> Co-occurrence studies like that conducted by <REF>Wettler and Rapp 1993</REF> have shown that for many purposes it is desirable to reduce the influence of word frequency on the co-occurrence counts .</S>
<S ID='S-29'> For the prediction of word associations they achieved best results when modifying each entry in the co-occurrence matrix using the following formula: Hereby </S>
</P>
<IMAGE ID='I-0'/>
<P>
<S ID='S-30'> is the frequency of common occurrence of the two words i and j, and f(i) is the corpus frequency of word i .</S>
<S ID='S-31'> However, for comparison, the simulations described below were also conducted using the original co-occurrence matrices </S>
</P>
<P>

<S ID='S-32'> Regardless of the formula applied, the English and the German matrix were both normalized .</S>
<S ID='S-33'> Starting from the normalized English and German matrices, the aim was to determine how far the similarity of the two matrices depends on the correspondence of word order .</S>
<S ID='S-34'> As a measure for matrix similarity the sum of the absolute differences of the values at corresponding matrix positions was used .</S>
<S ID='S-35'> This similarity measure leads to a value of zero for identical matrices, and to a value of 20 000 in the case that a non-zero entry in one of the 100 * 100 matrices always corresponds to a zero-value in the other .</S>
</P>
</DIV>
<DIV DEPTH='1' >
<HEADER ID='H-3'> Results </HEADER>
<P>
<S ID='S-36'> The simulation was conducted by randomly permuting the word order of the German matrix and then computing the similarity s to the English matrix .</S>
<S ID='S-37'> For each permutation it was determined how many words c had been shifted to positions different from those in the original German matrix .</S>
<S ID='S-38'> The simulation was continued until for each value of c a set of 1000 similarity values was available .</S>
<S ID='S-39'> Figure <CREF/> shows for the three formulas how the average similarity </S>
</P>
<IMAGE ID='I-1'/>
<P>
<S ID='S-40'> between the English and the German matrix depends on the number of non-corresponding word positions c .</S>
<S ID='S-41'> Each of the curves increases monotonically, with formula <CREF/> having the steepest, i. e. best discriminating characteristic .</S>
<S ID='S-42'> The dotted curves in figure <CREF/> are the minimum and maximum values in each set of 1000 similarity values for formula <CREF/> .</S>

</P>
</DIV>
<DIV DEPTH='1' >
<HEADER ID='H-4'> Discussion and prospects </HEADER>
<P>
<S ID='S-43'> It could be shown that even for unrelated English and German texts the patterns of word co-occurrences strongly correlate .</S>
<S ID='S-44'> The monotonically increasing character of the curves in figure <CREF/> indicates that in principle it should be possible to find word correspondences in two matrices of different languages by randomly permuting one of the matrices until the similarity function s reaches a minimum and thus indicates maximum similarity .</S>
<S ID='S-45'> However, the minimum-curve in figure <CREF/> suggests that there are some deep minima of the similarity function even in cases when many word correspondences are incorrect .</S>
<S ID='S-46'> An algorithm currently under construction therefore searches for many local minima, and tries to find out what word correspondences are the most reliable ones .</S>
<S ID='S-47'> In order to limit the search space, translations that are known beforehand can be used as anchor points .</S>
</P>
<P>
<S ID='S-48'> Future work will deal with the following as yet unresolved problems: Computational limitations require the vocabularies to be limited to subsets of all word types in large corpora .</S>
<S ID='S-49'> With criteria like the corpus frequency of a word, its specificity for a given domain, and the salience of its co-occurrence patterns, it should be possible to make a selection of corresponding vocabularies in the two languages .</S>
<S ID='S-50'> If morphological tools and disambiguators are available, preliminary lemmatization of the corpora would be desirable .</S>
</P>
<P>
<S ID='S-51'> Ambiguities in word translations can be taken into account by working with continuous probabilities to judge whether a word translation is correct instead of making a binary decision .</S>
<S ID='S-52'> Thereby, different sizes of the two matrices could be allowed for .</S>

</P>
<P>

<S ID='S-53'> It can be expected that with such a method the quality of the results depends on the thematic comparability of the corpora, but not on their degree of parallelism .</S>
<S ID='S-54'> As a further step, even with non parallel corpora it should be possible to locate comparable passages of text .</S>
</P>
</DIV>
<DIV DEPTH='1' >
<HEADER ID='H-5'> Acknowledgements </HEADER>
<P>
<S ID='S-55'> I thank Susan <REFAUTHOR>Armstrong</REFAUTHOR> and Manfred <REFAUTHOR>Wettler</REFAUTHOR> for their support of this project .</S>
<S ID='S-56'> Thanks also to Graham <REFAUTHOR>Russell</REFAUTHOR> and three anonymous reviewers for valuable comments on the manuscript .</S>
</P>
</DIV>
</BODY>
<REFERENCES>

<REFERENCE><SURNAME>Armstrong</SURNAME>, Susan; <SURNAME>Thompson</SURNAME>, Henry (<DATE>1995</DATE>) .<REFLABEL>Armstrong and Thompson 1995</REFLABEL>
</REFERENCE> A presentation of MLCC: Multilingual Corpora for Cooperation. Linguistic Database Workshop, Groningen.

<REFERENCE> Brown, Peter; Cocke, John; Della Pietra, Stephen A.; Della Pietra, Vincent J.; Jelinek, Fredrick; Lafferty, John D.; Mercer, Robert L.; Rossin, Paul <SURNAME>S</SURNAME>. (<DATE>1990</DATE>) .<REFLABEL>S 1990</REFLABEL>
</REFERENCE> A statistical approach to machine translation. Computational Linguistics, 16(2), 79-85.

<REFERENCE><SURNAME>Catizone</SURNAME>, Roberta; <SURNAME>Russell</SURNAME>, Graham; <SURNAME>Warwick</SURNAME>, Susan (<DATE>1989</DATE>) .<REFLABEL>Catizone et al. 1989</REFLABEL>
</REFERENCE> Deriving translation data from bilingual texts. In: U. Zernik (ed.): Proceedings of the First International Lexical Acquisition Workshop, Detroit.

<REFERENCE> <SURNAME>Church</SURNAME>, Kenneth W.; <SURNAME>Mercer</SURNAME>, Robert L. (<DATE>1993</DATE>) .<REFLABEL>Church and Mercer 1993</REFLABEL>
</REFERENCE> Introduction to the special issue on Computational Linguistics using large corpora. Computational Linguistics, 19(1), 1-24.

<REFERENCE> <SURNAME>Dagan</SURNAME>, Ido; <SURNAME>Church</SURNAME>, Kenneth W.; <SURNAME>Gale</SURNAME>, William <SURNAME>A</SURNAME>. (<DATE>1993</DATE>) .<REFLABEL>Dagan et al. 1993</REFLABEL>
</REFERENCE> Robust bilingual word alignment for machine aided translation. Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives. Columbus, Ohio, 1-8.

<REFERENCE><SURNAME>Kay</SURNAME>, Martin; <SURNAME>Roscheisen</SURNAME>, Martin (<DATE>1993</DATE>) .<REFLABEL>Kay and Roscheisen 1993</REFLABEL>
</REFERENCE> Text-Translation Alignment. Computational Linguistics, 19(1), 121-142.

<REFERENCE><SURNAME>Kent</SURNAME>, G.H.; <SURNAME>Rosanoff</SURNAME>, A.J. (<DATE>1910</DATE>) .<REFLABEL>Kent and Rosanoff 1910</REFLABEL>
</REFERENCE> A study of association in insanity. American Journal of Insanity, 67, 37-96, 317-390.

<REFERENCE> <SURNAME>Russell</SURNAME>, Wallace A. (<DATE>1970</DATE>) .<REFLABEL>Russell 1970</REFLABEL>
</REFERENCE> The complete German language norms for responses to 100 words from the Kent-Rosanoff word association test. In: L. Postman, G. Keppel (eds.): Norms of Word Association. New York: Academic Press, 53-94.

<REFERENCE><SURNAME>Wettler</SURNAME>, Manfred; <SURNAME>Rapp</SURNAME>, Reinhard (<DATE>1993</DATE>) .<REFLABEL>Wettler and Rapp 1993</REFLABEL>
</REFERENCE> Computation of word associations based on the co-occurrences of words in large corpora. In: Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives, Columbus, Ohio, 84-93.

</REFERENCES>
</PAPER>

Appendix E – Functional and Non-Functional Requirements

Functional Requirements

General

1. The system will be able to generate a set of slides from a source document. Each slide will be in the format of 'heading' followed by 'content/key terms'. The first slide will be an introductory slide displaying the document's title and author(s).

2. The system will return further research URLs relevant to the document from which a presentation is generated.

3. The system will support facility to save the state of any presentation.

4. The system will allow previous presentations to be loaded into the system.

5. The system will allow presentations to be exported to a proprietary Presentation Application.

6. The system will allow the user to change the quantity of key terms (% of identified key terms) to be returned.

7. The system will allow the user to alter the quantity of further research URLs to be returned as a result of generating a new presentation.

Navigation

8. The system will make visible to the user, what slide number is currently being viewed and the total number of slides.

9. The system will allow the user to navigate forwards and backwards through the slides, updating the current slide number being viewed.

10. The system will allow the current slide to be deleted, updating both the current slide being viewed and the total number of slides in the presentation as necessary.

11. The system will allow new slides to be appended to the end of the presentation. When a new slide is added, the current slide shown will change to the new slide and both the current slide number and total number of slides label will be updated.

Editing

12. The system shall allow users to paste text into the current slide window.

13. The system shall allow users to copy text in the current slide window.

14. The system shall allow users to cut text from the current slide window.

15. The system shall support the standard keyboard shortcuts for cut (ctrl + x), copy (ctrl + c) and paste (ctrl + v).

Non-Functional Requirements

16. The system will be compatible with the Linux Platform.

17. The system will as far as possible utilise domain independent techniques.

18. The system will be menu driven with additional navigation buttons, laid out in a user-friendly manner conforming to a set of heuristics/guidelines.

19. The system will be sufficiently robust. Any errors in executing the operations of the system should be communicated to the user through appropriate, visible feedback.

Appendix F – System Processes

Name	Description	Tools/Languages Required
Tagger	Utilises the Unigram and Brill Taggers of the NTK library to tag an untagged corpus and document.	NLTK, Python
Chunking	Utilises NLTK's chunker, searching for Noun Phrases.	NLTK, Python
Ranking (TF.IDF)	Takes the list of identified Noun Phrases from the chunker operation and builds up an inverted index and a separate index for a source document. Uses TF.IDF to compute all terms whose score is above some threshold.	Python
Segmenting	Takes Key Terms identified through Ranking operation and segregates the terms into lists (slides) based upon which subheading in the document they appear in.	Python
Relevant Further Research URLs	The top three highest scoring TF.IDF terms are processed through the Google API to find URLs of similar research material.	Google API, Java
Save Presentation Content	Allows the current presentation being viewed/edited to be saved for subsequent retrieval. A special character(s) is placed in the file at various points to signal the break point between slides.	Java
Load Presentation Content	Previously saved presentations are read into the system, searching for the special character(s) to signify the partitioning into slides.	Java
Export Presentation	The current presentation being viewed is exported into OO.org's Presentation Application (Impress). Several XML files conforming to Impress's DTD scheme are created. These files are then zipped and the single file is converted to Impress's file extension i.e. '.sxi'.	Java

Appendix G – Results for Lower and Upper bounds

Human Judged Key Terms		Lower Bound		Upper Bound	
Document (.sent)	Group Size	precision	recall	precision	recall
9407001	10	0.1	0.1	0.5	0.5
9505004	10	0	0	0.5	0.5
9505007	10	0	0	0.3	0.3
9505008	10	0	0	0.5	0.4
9505009	10	0	0	0.3	0.3
9505010	10	0	0	0.5	0.5
9505011	10	0	0	0.6	0.8
9505024	10	0	0	0.4	0.3
9505025	10	0	0	0.3	0.3
9505028	10	0.1	0.1	0.6	0.6
9505030	10	0	0	0.6	0.5
9506025	10	0	0	0.6	0.5
9508005	10	0	0	1	1
9509001	10	0	0	0.5	0.5
9512003	10	0	0	0.8	0.7
Average	-	0.013	0.013	0.533	0.513

Appendix H – Instruction Manual

Instruction Manual - Table of Contents

Open Lecture Content Generator

To open the program, at command line (prompt), change to the directory containing all the .py and .java files. Type:

java Gui

Of limited use, to view only all the key-terms of a source document without needing to open the program, change to the directory containing all the .py and .java files. Type:

./lectureterms.py <source document> 10

The sequence of operations the system performs in generating the system is outputted at command line. The operation ends by outputting all the key-terms, ranked in order of relevance to the source document.

Main Window View

The Main view of the system is divided into three broad areas, the menus toolbar, the slide content and further research boxes, and the navigation buttons.

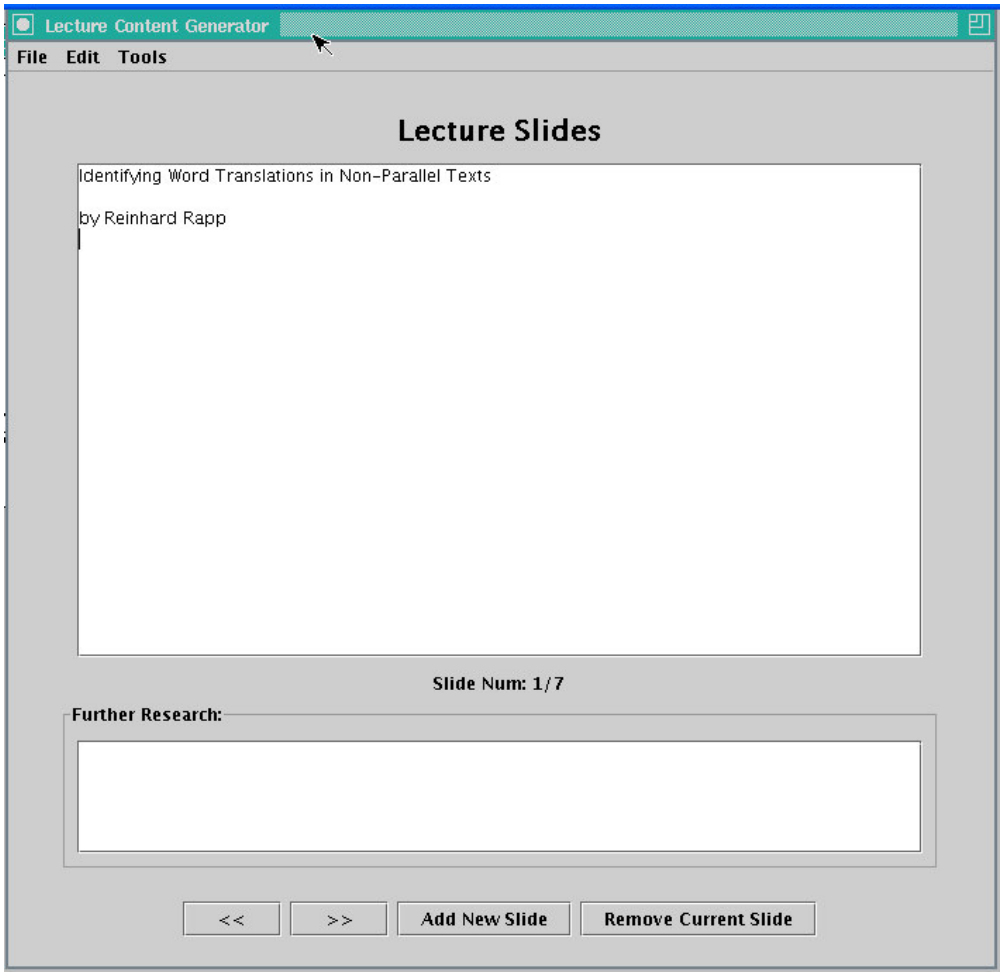


Figure 1.1

The toolbar consists of three menus from which the main functionality of the system can accessed. The 'File' Menu offers the options of creating a new slideshow from a source document, loading/saving a presentation and exporting the presentation. The 'Edit' Menu supports common editing features for modifying the content of the lecture slide area. The 'Tools' Menu, allows a user to set preferences with respect to creating a new slideshow.

The main text area is known as the lecture content/slide area. The contents of the current slide is displayed within this text area. Directly below this text area is an indicator to both the current slide being viewed and the total number of slides. The number preceding the '/' indicates the slide number being viewed,

while the number proceeding the '/', indicates the total number of slides in the current presentation.

The other text area, 'further research', is utilised when creating a new presentation to offer links (URLs) to related web resources.

Navigating around the slides in the presentation is achieved through the '<<' and '>>' buttons at the bottom of the window. The two adjacent buttons in addition, allow slides to be created and removed.

Section 1: New Presentation

In creating a new presentation, an XML source document within in the field of science (preferably within the field of Natural Language Processing) is required. The document must conform to the DTD scheme found in Appendix C.

To create a new presentation from a source document:

1. Choose the 'New' option from the 'File' Menu.
2. From the open dialog box, choose the source document and select 'Open'. A message box will inform that the operation is progress. After around 1-2 minutes, the message box should disappear and both the content of the lecture slides and a list of further research weblinks will appear.

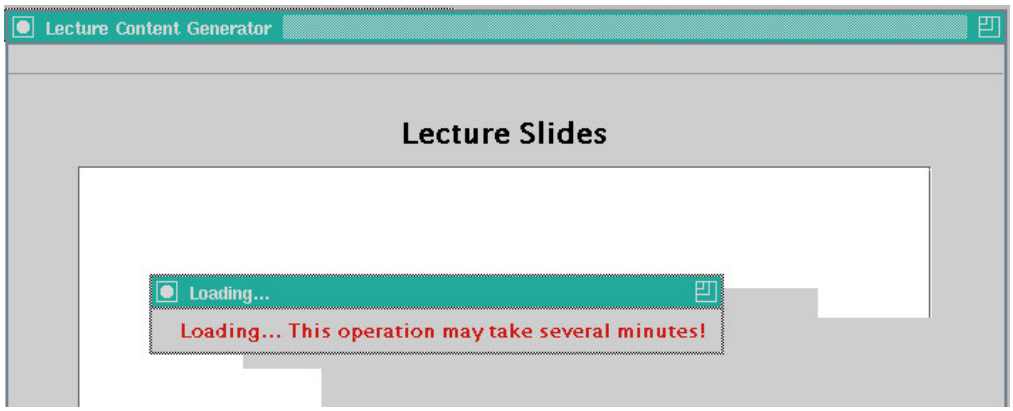


Figure S1.1

To view any of the related weblinks:

1. Highlight a URL in the further research box.
2. Press 'Ctrl + c' or select the 'Copy' option from the 'Edit' Menu.
3. Open a web browser and paste the link into the location bar.

Section 2: Slideshow Navigation

The buttons at the bottom of the window are used for navigating around the presentation. To move forward one slide, press the '>>' button. To move back one slide, press the '<<' button. Below the lecture text area, the number preceding the forward slash ('/'), will update to the current slide being viewed.

To add a new slide, click the 'Add New Slide' button. This will append a new slide to the end of the presentation. The new *blank* slide will be displayed.

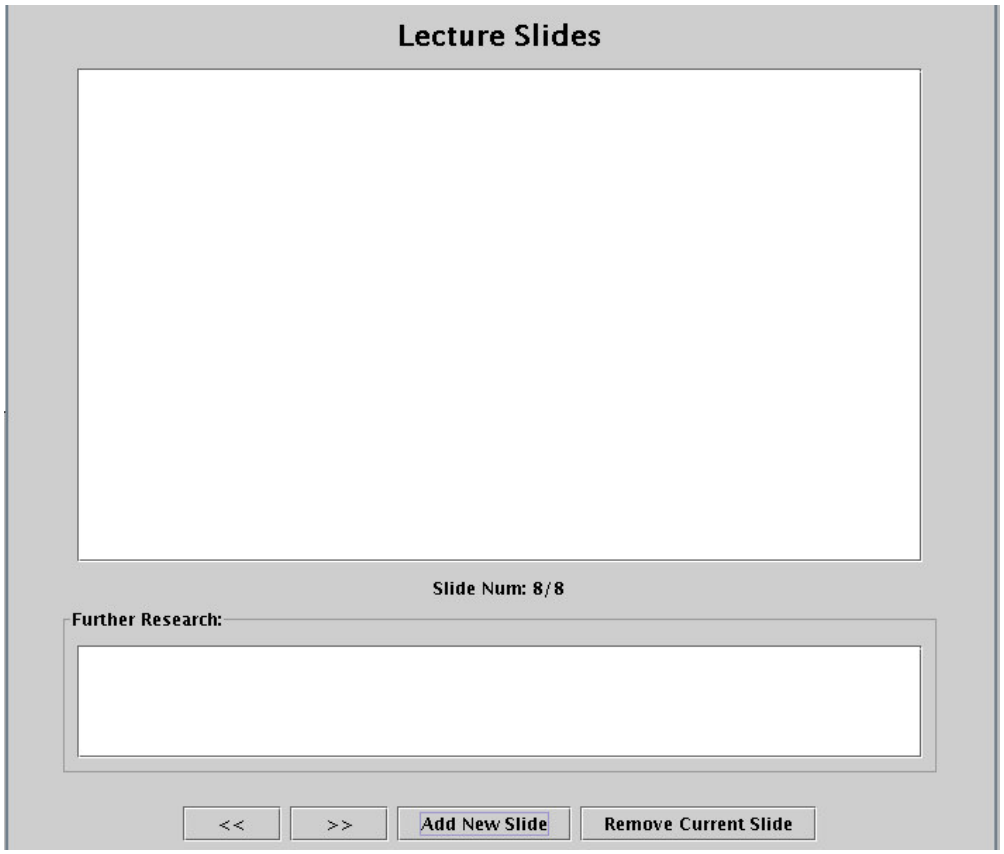


Figure S2.1

To remove a slide:

1. Navigate to the slide to be deleted.
2. Click the 'Remove Current Slide' button. Unless the last slide of the presentation is selected, the total number of slides is reduced by one and the slide displayed changes to the proceeding slide. If the last slide of presentation is deleted, the current slide changes to the preceding slide.

Section 3: Open Presentation

To open an existing presentation:

1. Choose the 'Open' option from the 'File' Menu.
2. Choose a presentation ('.lct' file) from the open dialog box and select 'Open'. A confirmation box should appear to confirm that the file has successfully loaded.

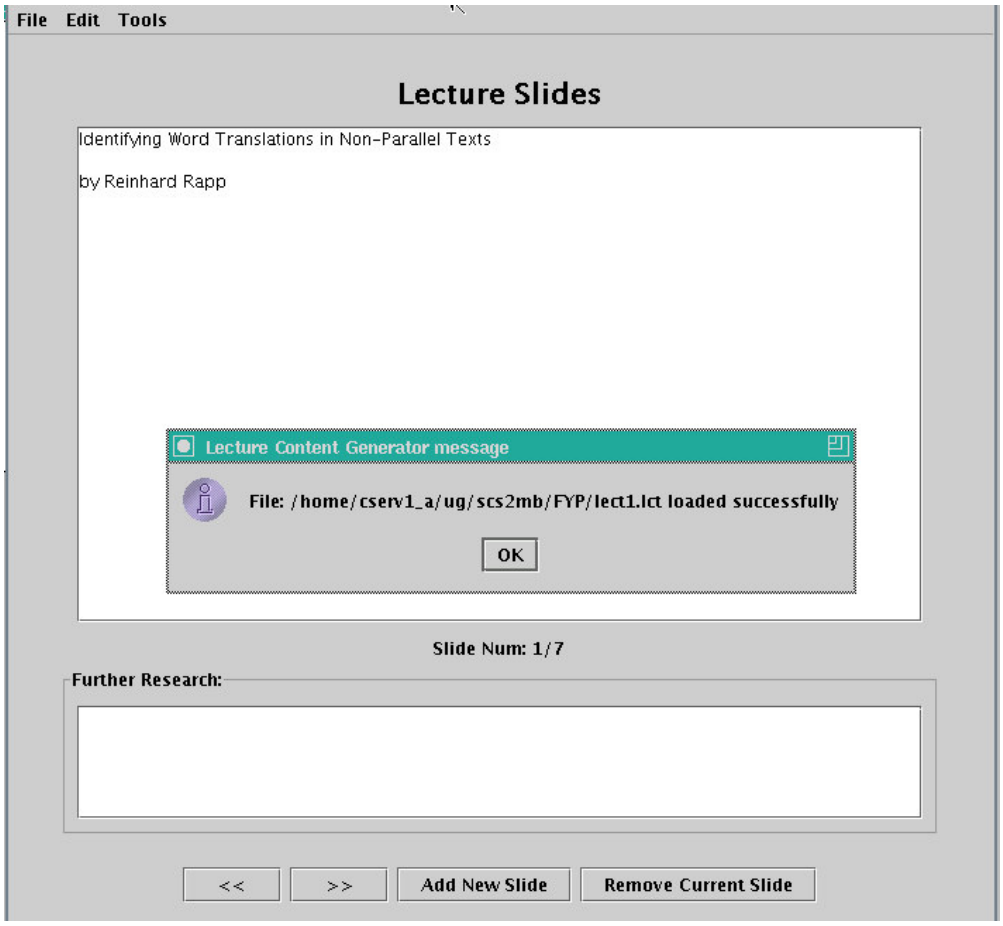


Figure S3.1

Section 4: Save Presentation

To save the current presentation:

1. Choose the ‘Save’ option from the ‘File’ Menu.
2. Type the filename or select a file to be saved from the save dialog box and click on the ‘Save’ button. A confirmation box should appear to confirm that the file has successfully saved.

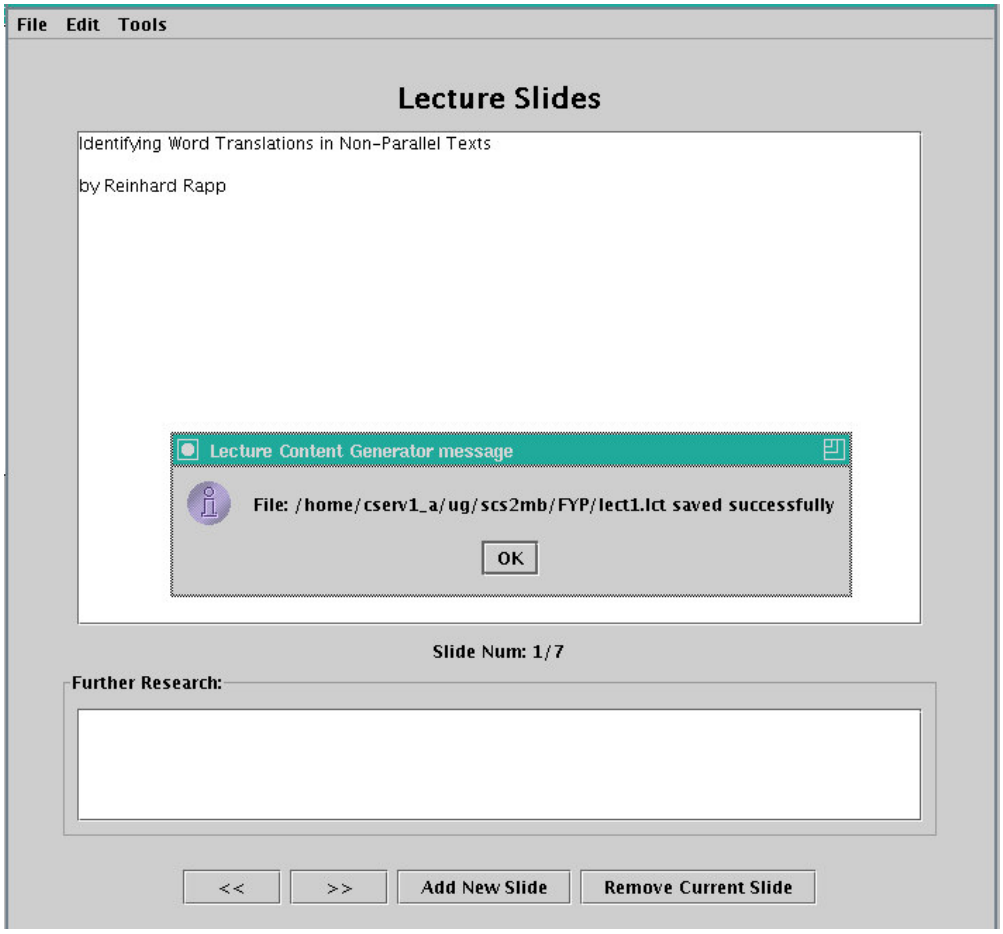


Figure S4.1

Section 5: Export Presentation

To export the current presentation to Open Office's presentation application (Impress):

1. Choose the 'Export' -> 'Open Office' option from the 'File' Menu.

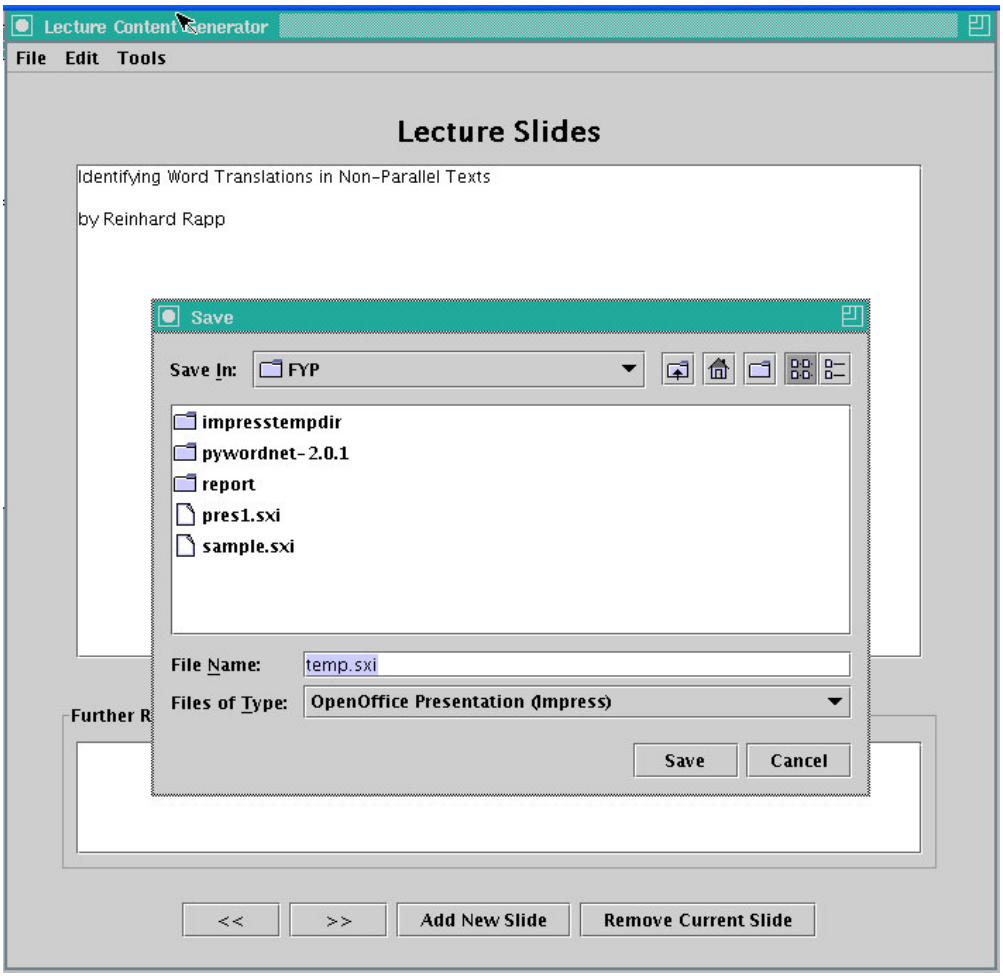


Figure S5.1

2. Type the filename or select a file to be saved from the save dialog box and click on the 'Save' button. A confirmation box should appear to confirm that the file has successfully saved.

Section 6: Editing Features

The editing features supported are ‘undo/redo’, ‘copy’, ‘cut’, ‘paste’. All editing operations apply only to the slide text area except for the copy function, which equally applies to the further research text area.

To Undo or Redo a modification to the content of the current slide, select as appropriate, the ‘Undo’ or ‘Redo’ option from the ‘Edit’ Menu. The exact effect of selecting undo and redo is further qualified, as the adjacent word to both options, states the category of modification e.g. ‘Addition’.

To cut text from the lecture slide:

1. Highlight the text to be cut in the lecture slide.
2. Either press ‘Ctrl + x’ or select the ‘Cut’ option from the ‘Edit’ Menu.

To copy text from the lecture slide:

1. Highlight the text to be copied in the lecture slide.
2. Either press ‘Ctrl + c’ or select the ‘Copy’ option from the ‘Edit’ Menu.

To paste text from the lecture slide:

1. Place the cursor to the point in the lecture text area where the text is to be paste.
2. Either press ‘Ctrl + v’ or select the ‘Paste’ option from the ‘Edit’ Menu.

Section 7: Preferences

To change the settings for creating new slideshows, choose the ‘Preferences’ option from the ‘Tools’ Menu. The preferences allow the quantity of key-terms and number of further research results to be altered.

To change the percentage of identified key terms to be returned for a given presentation, move the slider as appropriate.

To change the quantity of further research results to be returned, select from the ‘top 3’, ‘top 5’ and ‘top 10’ options in the drop-down box.

Choose the ‘OK’ button to save any changes made.

Appendix I – Guidelines for Human Judge Test

Term Extraction Evaluation Guidelines

The purpose of this evaluation is to determine the accuracy of the term extraction system. You will be required to review 15 scientific articles and in each case identify 10 key terms which best characterise the document.

Please read carefully the following instructions:

1. Browse to the following directory: '/home/csunix/nlplib/rhetstatus/unseen/text/'. You will find in this directory a repository of scientific articles. The specific documents, which you should use/open to carry out this test, are:

9407001.sent, 9505004.sent, 9505007.sent, 9505008.sent, 9505009.sent, 9505010.sent, 9505011.sent, 9505024.sent, 9505025.sent, 9505028.sent, 9505030.sent, 9506025.sent, 9508005.sent, 9509001.sent, 9512003.sent

2. Create a new document in .txt format and then for each document above, record:

(i.) The File Name of the document
(ii.) The key-terms, which best characterise the document. This requires you to read the document carefully and identify 10 'Noun Phrases', which in your opinion, best reflect the document's contents. By Noun Phrases, that is any sequence of words made up from determiners, adjectives and all noun derivations. The identified key-terms can be both single words and terms (multi-word phrases), but in the case of single words, only nouns should be identified, not determiners or adjectives. The first word of multi-word terms, should not be a determiner. Examples of the types of POS structure of key-terms that should be found include:

‘term extraction’, ‘co-occurrence counts’, ‘lexical statistics’, ‘probability’, ‘OCR programs’, ‘aligned’, ‘unconstrained model’, ‘context free grammar’.

3. E-mail your completed .txt document as an attachment to: scs2mb@comp.leeds.ac.uk

Thank you for giving up your valuable time to assist in this evaluation.