University of Southampton
School of Electronics and Computer Science
Faculty of Engineering, Science and Mathematics
Southampton SO17 1BJ

Automatic Key Theme Extraction in Natural Language Texts

by

Bon Yiu Mo

A thesis submitted for the award of Doctor of Engineering (EngD)

September 2005

Academic Supervisor: Professor Wendy Hall
Industrial Supervisor: John Darlington

Active Navigation

Abstract

Information is the most powerful resource available to an organisation. Problems arise when the amount of management needed to effectively organise the mass of information available in data-repositories starts to increase, and reaches a level that is impossible to maintain. Instead of providing value to an organisation the information can serve to confuse and hamper. This research presents the topic of automatic key theme extraction as a method for information management, specifically the extraction of pertinent information from natural language texts. The motivation for this research was to achieve improved accuracy in automatic key theme extraction in natural language texts. The performance was evaluated against an industrial context, which was provided by Active Navigation Ltd, a content management system. The author has produced an architecture for theme extraction using a pipeline of individual processing components that adhere to a lossless information strategy. This lossless architecture has shown that it is capable of providing a higher accuracy of extracting key themes from natural language texts than that of Active Navigation. The accurate extraction of key themes is essential as it provides a solid base for other Active Navigation information navigation tasks. These include advanced search, categorisation, building summaries, finding related documents, and dynamic linking. Improving these navigation techniques increases the effectiveness of the content management system.

Acknowledgements

The author would like to acknowledge the direction, insight and inspiration provided by both John Darlington and Professor Wendy Hall, his industrial and academic supervisors respectively.

John has been the driving force for this thesis and his knowledge and experience in this area of research has been invaluable. Furthermore he has been instrumental in laboriously educating the author in this challenging work. Wendy has provided an air of authority and wisdom that has been enormously beneficial to the author throughout his study.

The author would also like to express thanks to his colleagues at Active Navigation who have all contributed to a warm working environment and aided in the progression of the thesis.

Finally, the author must express gratitude to his family and friends who have served to motivate and provide support. Thank you.

Contents

List of Equations

List of Figures

List of Tables

1. Introduction

Information is the most powerful resource available to an organisation. To maximise the value of information, people must get access to all and only the information they need, when they need it [O'Hara, 2002]. Problems arise when the amount of management needed to effectively organise the mass of information available in data-repositories starts to increase, and reaches a level that is impossible to maintain. Instead of providing value to an organisation the information can serve to confuse and hamper.

This chapter presents the topic of automatic key theme extraction as a method for information management, specifically the extraction of pertinent information from natural language texts. The traditional methods for information management are also discussed and their problems highlighted.

The research objectives for this project are stated, which include the motivation, goals and deliverables for the system. The chapter concludes with a structure for the rest of the thesis.

1.1. Information Management

1.1.1. Current Approaches to Information Management

The two common approaches to information management are categorisation and search:

Categorisation – This is the classification of documents based on key-words. Traditional manual categorisation is suitable for shallow taxonomies or for a small quantity of documents, but is inefficient for large corpora. Large-scale manual categorisation usually leads to difficulties in categorising new documents, and deeply embedded documents can be difficult to find. Categorisation of large corpora needs to be automated which requires documents to be automatically classified by their key content.

Search – This is the retrieval of documents based on key-words. Traditional key-word search engines are very effective at filtering pages that match explicit queries. Unfortunately most people find articulating what they want extremely difficult, especially if forced to use a limited vocabulary such as key-words [Middleton et al., 2001]. For

example a document has the phrase 'a green car at red lights', a search for the words 'red' and 'car' retrieves the document as a positive result, even though the entity 'red car' does not exist in the document, which is poor precision on the part of the search engine. In essence key-word search engines cannot acquire the same context that people can because information is too abstractly represented in a document.

Both approaches to information management lack the capability to automatically recognise the key content of a document, and to perform the categorisation or search on this key content.

1.1.2. Automatic Key Theme Extraction

The primary requirement to improve information management is by using information context. This means extracting the key content from a document, referred to as the 'themes', and not just by the key-words.

A theme can be a concept, a description or a semantic notion. A theme can be either a single word or a phrase. A theme can be a keyword, however they tend to be richer as they can contain information context to form phrases. The core requirement of a theme is to represent a document. This requirement does not change over time. Therefore the job of the theme extraction process is to produce the best representation of the document.

Examples of themes are illustrated in Figure 1-1 which is an example of Active Navigation's theme extraction and navigation. In Figure 1-1 the themes of interest are marked-up as links in the document. The clicking of a theme will reveal links to other documents sharing similar themes. For example, the theme 'ARPA Internet', links to documents that have themes about 'ARPA Internet' and 'Internet'.

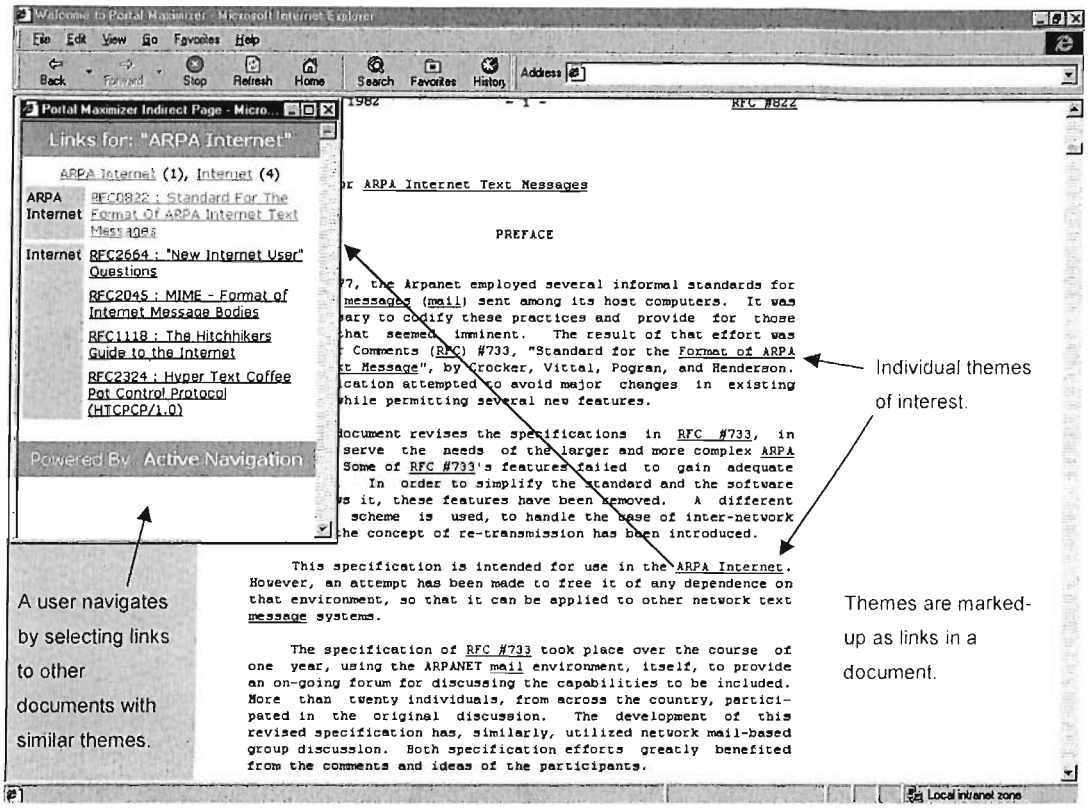

Figure 1-1 – An illustration of themes

The theme extraction process is non-trivial due to the complexities of natural language. The process of extracting themes in a document is a high-level classification problem that people take for granted because of our extraordinary data processing abilities.

Whilst people typically have the cultural and linguistic experience to comprehend a notion such as a theme, a computer system requires a considerable amount of predefined knowledge to perform the same task.

What is required is a process that automates the human ability to extract the key themes in a document and not just the key-words. This process is termed 'automatic key theme extraction'.

To simulate this process computationally requires an understanding of natural language, in particular the grammar and ambiguity of a language, and to provide a suitable language model to extract the information with.

There have been serious efforts in the area of Natural Language Processing to automatically extract themes in documents by means of linguistic methods based on language parsing. However, despite these efforts this approach still lacks robustness and more importantly has serious performance problems.

This research attempts to bridge the gap between the capabilities of humans and computer systems to automatically extract key themes from natural language documents.

1.1.3. Natural Language Texts

Information repositories are generated on a daily basis. Most information in repositories is stored electronically in the form of documents. However, rarely does the storage of documentation come in a standard or structured format. How a document is written can be dependent on an individuals writing style, use of grammar and extent of their vocabulary.

Documents are a stream of words and are difficult to interpret because of the ambiguous nature of language. Single words, phrases and even sentences can have ambiguous meaning. A theme extraction system must be able to resolve these ambiguities.

These complications mean that a document's key themes can be difficult to comprehend, because they are not expressed clearly. The challenge of natural language texts ensures problems when attempting to interpret the meaning of the document and extracting only key themes from it.

1.2. Research Objectives

1.2.1. Motivation

The motivation for this research is to achieve improved accuracy in automatic key theme extraction in natural language texts. The performance is evaluated against an industrial context, which is provided by Active Navigation Ltd[1], a content management system (CMS).

This evaluation benefits Active Navigation as it shows how to improve their theme extraction, which in turn creates a better CMS. This improved CMS allows Active Navigation to provide more efficient management and organisation of data-repositories for its clients. A consequence of this is that a client requires less human intervention to manage their information and should result in a reduction in the quantity of manual labour and ultimately cost.

1.2.2. Goals

The success criterion is to produce a useable grammar and language model that can be implemented into Active Navigation's current software. It is assumed that the research extends the current software's capabilities by providing a more accurate navigation experience of documents.

A new approach to theme extraction requires focus on many issues of computational linguistics and in particular the problems of grammar, ambiguity and the choice of language model.

This requires analysis and development of techniques for theme extraction. It attempts to resolve some of the complexities of theme extraction in natural language by using an improved language model.

[1] http://www.activenavigation.com – Active Navigation Ltd was the author's sponsoring company

It is believed that Natural Language Processing techniques can help to derive better representation of documents than any key-word methods commonly used in statistical document analysis.

This is based on the premise that linguistic processing can uncover certain critical semantic aspects of document content, something that simple word counting cannot do, and this should lead to a more accurate representation.

1.2.3. Deliverables

The first deliverable is an architecture for theme extraction using a pipeline of individual processing components that adhere to a lossless information strategy.

At the heart of the architecture is the idea of minimising the information loss. The extraction of themes to represent a document results in high information loss as only the themes are kept. To minimise this loss the strategy used is breadth-first non-determinism. This strategy will be referred to as minimising information loss in this thesis.

A lossless architecture means that all possibilities are tried rather than just a subset. It is an exhaustive, brute-force approach with no pruning. This is strictly for academic gain as in the real-world this is never likely to happen as it is too computationally expensive. There are time considerations to take into account when using a brute-force approach and this rarely fits into company strategies.

Information persistence between components is required so that all information is preserved. This contributes to other desired attributes such as minimising error propagation, and improving the accuracy of components because the information is handled more transparently.

A novel formalism is required to represent information persisted by each component. This formalism is called a 'quad' and it captures attribute-value pairs, the confidence value associated to the pair, and provenance information which stores the origins of a quad.

The lossless architecture keeps track of all choices made by the components using these quads.

The aim of the architecture is to accurately extract key themes from natural language documents using a fusion of linguistic, statistical and grammar rules.

The accurate extraction of key themes is essential as it provides a solid base for other Active Navigation information navigation tasks. These include advanced search, categorisation, building summaries, finding related documents, and dynamic linking.

1.3. Structure of the Thesis

Chapter 2 provides a background to Active Navigation and in particular its branding, method and competitors. It also contains the motivation for using themes and covers important aspects of themes such as their purpose, theories, extraction, and ranking.

Chapter 3 provides a background to Natural Language Processing and in particular the contrast and similarity of the two language models of constituency and dependency which all Information Extraction systems are based. The chapter also provides the different types of parsers used for both language models and also example benchmark systems.

Chapter 4 raises the issues for both language models, which includes an analysis and discussion of the sub-standard components in a constituency model, and the use of these constituency components and problems of dependency structure parsing in a dependency model. The choice of grammar with which to map a language is also discussed.

Chapter 5 outlines the design of the automatic key theme extraction system. It begins with an overview of traditional architecture issues and a justification for a lossless architecture which is the basis of a new system. The quad formalism is then presented as the information representation mechanism in the system. This is followed by a detailed presentation of all the components used in the lossless architecture, which starts at the input document and finishes with the key themes extracted for that document. Then the issue of computational feasibility of a lossless architecture is raised. The chapter concludes with an overview of the system implementation.

Chapter 6 provides an evaluation of key theme extraction of the system compared with Active Navigation. This includes a history of Information Extraction evaluation, and the scoring strategy and metrics used. The results of the evaluation and a discussion are provided. The chapter concludes with a summary of the results.

Chapter 7 provides the conclusions for the system. It provides a summary of the thesis, whether the system has met the research objectives, and how the research can improve Active Navigation. The final section outlines future work to improve key theme extraction, how these key themes can be applied, and looks at the Semantic Web.

2. Background of Active Navigation and Themes

2.1. Active Navigation Ltd

This section provides an overview of Active Navigation which is a content management system and also the author's sponsoring company. It outlines how the system is branded in the industrial market place and provides a description of the techniques that it uses as a navigation solution for information management. This section compares and contrasts a couple of the company's key competitors, which shows different approaches to a solution for the information management problem. This section concludes with a summary.

2.1.1. Branding

The quantity of information that needs to be managed is growing quickly. This explosion of unstructured information has made traditional approaches ineffective. Companies cannot afford the time and resources of manually categorising information. Existing search engines and information management systems typically require an unacceptable amount of human intervention or generally produce an inefficient and ineffective user experience.

The result is that companies are struggling to organise and manage information as it is created and collected within their enterprise, or published on their public websites.

Information management demands a fresh method for managing unstructured information. Active Navigation provides a complete, intuitive, navigation solution that guides non-expert users to the valuable information they need by intelligently managing unstructured information.

Active Navigation emerged as a commercial vehicle for research at the University of Southampton into information navigation. Since then it has focused on the analysis of text using linguistic and statistical techniques.

The result is the most advanced technology available commercially for the rapid and accurate analysis of large quantities of information[2].

2.1.2. Method

2.1.2.1. Navigation Techniques

Active Navigation uses four main navigation techniques to provide a complete navigation solution these are categorisation, search, related articles and dynamic links.

- *Categorisation* – This is the classification of documents based on themes. Active Navigation's categorisation feature helps an information specialist to create a taxonomy that reflects the information content. The taxonomy is then filled with documents either automatically or semi-automatically with the information specialist making suggestions. This process builds taxonomies within a fraction of the time taken using traditional approaches.

- *Search* – This is the retrieval of documents based on themes rather than key-words. Most search engines retrieve documents based on occurrences of particular key-words. However, this method lacks any sense of context which is why many search results are spurious. Active Navigation maintains context by searching for themes contained within documents. Searching by themes ensures that only relevant results are returned.

- *Related Articles* – This uses all the themes of a document to search for similar documents. Typically, related articles are created manually by a team of editors creating links between related documents. Active Navigation automatically generates a list of related documents that contain the same themes as the document being viewed.

- *Dynamic Links* – This is the navigation of documents by a particular theme. Dynamic links are a unique selling proposition of Active Navigation and their importance is highlighted in the next section.

[2] http://www.activenavigation.com/Solutions/solutions.htm

2.1.2.2. Dynamic Links

Standard hyperlink injection has a variety of problems as it imposes a rigid structure on how users are able to navigate information irrespective of the needs of the user. Hyperlinks only provide a one-to-one and mono-directional relationship between documents, and tend to be very subjective in nature. A web author manually inserts links into documents which tends to be a time and resource intensive task. Poor link maintenance can result in broken links and ultimately the loss of customer loyalty. Hyperlinks are time consuming, expensive to maintain and are inefficient with increasing content.

The Active Navigation approach is to use a link database or 'linkbase' which is metadata about a collection of documents that contains a list of themes and documents, and the relationships between them, which is often many-to-many. The list of themes is created by theme extraction over the documents. The linkbase is then used to provide dynamic link injection which automatically populates a document with dynamic links which, when navigated, reveal a set of related documents based on the theme.

This Active Navigation approach originated from ideas presented by Microcosm [Davis et al., 1992a, 1992b, 1993] and later by Distributed Link Service (DLS) System [Carr et al., 1995, 1998].

Microcosm was an Open Hypermedia System, in these systems links are first-class objects, stored and managed separately from multimedia data [Carr et al., 2001]. Link information was stored separately from documents in linkbases. This storage reduced link maintenance as it allowed changes made to a linkbase to be immediately effective wherever the link was available. El-Beltagy [El-Beltagy et al., 1999] mentioned that the concept of abstracting links from documents allowed for a great deal of flexibility since it allowed adding hypermedia functionality to documents without changing the document format or embedding mark-up information.

Microcosm allowed the user to navigate through large bodies of information by a number of different layers of link mechanism. In most hypermedia systems, links have specific source and destination anchors. Microcosm supported such 'specific' links, but also supported more general links. 'Generic' links provided a link from a particular object at

any position in any document that connects to a particular object in a destination document [Davis et al., 1992a, 1992b]. The generic link enabled the destination of a link to be resolved at run-time calculated on the basis of the content of a source anchor rather than simply its location in a document.

The DLS provided a powerful framework to aid navigation and authoring [Carr et al., 2001]. It was based on Microcosm, and similarly it utilised a variety of link database processes to offer flexible hypertext functionality to a wide range of end-user applications, by providing an independent system of link services for the World Wide Web (WWW) and allowed authors to create configurable navigation pathways for collections of WWW resources. This was achieved by adding links to documents as they were delivered from a WWW server, and by allowing the users to choose the sets of links that they saw according to their interests [Carr et al., 1998].

This link service provided important functionality for information systems by increasing the navigation options for users without increasing the problems of information maintenance. In conjunction with the WWW it provided a powerful tool with which to address the restrictions of embedded links that are fixed with the publication of a document [Carr et al., 1998].

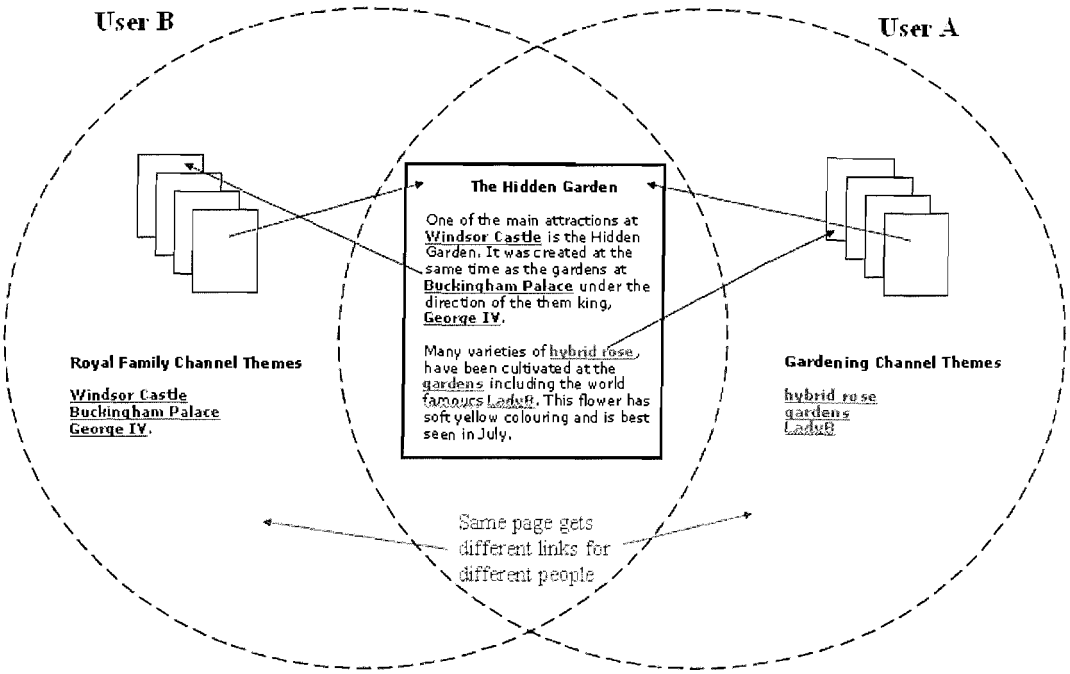

Figure 2-1 – Dynamic link generation

In the Active Navigation system, each user can see different dynamic links depending on which linkbases are active. This means that installing different linkbases, as appropriate, it is possible to provide different views on the same set of documents [Davis et al., 1993].

In the document 'The Hidden Garden' (see Figure 2-1) the themes of interest for a user are marked-up as dynamic links. These links are used to navigate between related documents. For example, User A is interested in themes based on 'gardening' whilst User B is more interested in themes concerning the 'Royal Family'. The dynamic links are generated on a particular document based on a user's themes. So themes such as 'hybrid rose' and 'gardens' become links for User A from the document 'The Hidden Garden' but themes such as 'Windsor Castle' and 'George IV' become links for User B.

All the navigation techniques central to the navigation solution require understanding of the language in order to build relationships between documents. The key requirement therefore is the extraction of the key themes from a document.

2.1.3. Purpose of a Theme

The primary requirement to achieve a navigation solution is to provide better navigation by using information context. This means extracting the key content from a document, referred to as the themes, and not just by the keywords. A theme in this context can be a concept, a description or a semantic notion.

Active Navigation's method is to extract these themes then to link together similar documents based on these themes.

In an information management system such as Active Navigation, the information held by the system is not the actual document but some metadata representation of the document. Within such a system a document is represented by an object. A document object consists of both hidden and visible information. The hidden part is the text of the actual document that is not held in the system, and it is up to the theme extraction process to make visible the important parts of the document by exposing the key themes.

The purpose of a theme is to represent a document. This purpose does not change over time. Therefore the job of the theme extraction process is to produce the best representation of the document.

The theme extraction and link generation should be viewed as separate components. A link is independent to a theme, which itself is independent of a document. All three entities are separate from one another.

Creating the links between documents requires a representation of the potential interests of the user. An interest is represented by themes in some source document. The themes can be as simple as a text string or as complex as a semantic relationship. The links must have destination anchors so the set of potential target documents are identified.

These elements are used in a matching function that provides links as output. The matching function creates links between all the documents based on the extracted themes. The matching function can utilise many layers of information, currently the level of detail used at Active Navigation is a string match, but this lacks context. A superior matching function requires augmentation with information such as an ontology or another

information representation. The representation of a theme can be enhanced by using a higher layer of information.

The matching function can create links based on real world knowledge, but does not currently do so. For example 'metal door' is a theme and it can have links to other themes such as 'aluminium door' and 'steel door'. The matcher can take into account world knowledge that both 'aluminium' and 'steel' are both forms of metal. When a user selects a theme, potential documents that share similar themes become available for navigation.

In summary the themes must be expressed effectively otherwise the matching function is obsolete.

2.1.4. Competitors

Active Navigation's main competitors use text analysis techniques that range from pure statistical to pure linguistic. The systems that use linguistics require human language experts for producing lexical semantic tools such as language or domain specific thesauri or lexicons, and to perform tasks such as manually predefining rules and manual classification.

Most of the competitor systems require training. This is typically used for statistical classification and can be unsupervised or supervised. Unsupervised training requires an existing document collection. Supervised training needs more human effort as it requires both positive and negative exemplar documents. Both automatically generate rules that define categories.

The weaknesses that most systems have are domain dependent navigation techniques, manual tuning and manual training of systems.

The next two sections give a brief overview of two major competitors, Autonomy[3] and Content Analyst[4]. This is followed by a quick assessment of the systems.

[3] http://www.autonomy.com

[4] http://www.contentanalyst.com

2.1.4.1. Autonomy

Autonomy's technology brings a fully-automated solution to the problems caused by the exponential growth in unstructured information. By facilitating the automated management and dynamic personalisation of information, it enables enterprises to optimise both the potential and utilisation of their corporate intranets, extranets and commercial websites.

Autonomy develops infrastructure technology that automatically processes and organises large amounts of unstructured information into personally relevant content in real-time and in an efficient manner. The core technology an Intelligent Data Operating Layer (IDOL) provides a platform for the automatic categorisation, hyperlinking, retrieval and profiling of unstructured information, thereby enabling the automatic delivery of large volumes of personalised information.

Autonomy's IDOL brings together the works of Thomas Bayes and Claude Shannon. Thomas Bayes' work uses statistical calculations of the relationships between multiple variables and the impact that one variable has on another. A variable in this context is a piece of information, and the relationship it forms with other pieces of information determines its meaning. Claude Shannon's contribution is a theory that the rarer a word or phrase, the more information it conveys. This approach is independent of the language of the text and allows the main concepts to be identified and prioritised. It does so by identifying patterns of words and ideas, as well as determining their relative importance to individual users.

Autonomy's main information management tasks are listed below:

- *Automatic Hyperlinking* – Autonomy's infrastructure identifies vital relationships between information enabling the cross-referencing of content. Given any document or set of documents, Autonomy can identify related material within the operating layer. Autonomy's IDOL generates hyperlinks in real-time, ensuring they are immediately up-to-date. Links are automatically inserted at the time a document is viewed.

- *Automatic Contextual Summarisation* – Autonomy returns a summary of the information containing the most salient concepts of the content. Summaries can be generated that relate to the context of the original inquiry – allowing the most applicable dynamic abstract to be provided for a given operation.

- *Conceptual Search* – Built on a unique pattern-recognition technology, Autonomy's core engine enables a manual or fully automated precise means of matching and identifying the similarity of pieces of information.

- *Automatic Categorisation* – Autonomy's technology can automatically categorise data. The flexibility of Autonomy's categorisation features allows you to precisely derive categories using concepts found within unstructured text. This ensures that all data is classified in the correct context with the utmost accuracy.

- *Automatic Taxonomy Generation* – Autonomy can automatically and consistently understand and create deep hierarchical contextual taxonomies of information based on conceptual understanding.

Autonomy's method is based on statistical modelling, which allows the information content to be domain and language independent. This also means that linguistic tools such as lexicons and grammar are not required. The IDOL treats words and phrases as abstract symbols of meaning deriving its understanding through the context of their occurrence. Statistical modelling means that human intervention is made redundant.

2.1.4.2. Content Analyst

Content Analyst technology is a powerful example of a new class of technologies known as Text Analytics designed to transform large volumes of unstructured data into relevant actionable information. Text Analytics products automate most of the human activity traditionally associated with understanding, organising, prioritising and retrieving information from large sources of unstructured data.

Content Analyst gives organisations the ability to quickly analyse, organise, access and share information across multiple languages without extensive human intervention.

Content Analyst's method is Latent Semantic Indexing (LSI). LSI is a machine learning technique that extracts and compares every contextual relation among every concept, in every document, within a collection of documents. It was originally developed by Science Applications International Corporation (SAIC).

LSI has been designed to solve two fundamental problems of natural language: synonymy (where multiple words have the same meaning), and polysemy (where the same word has multiple meanings). For example in the context of key-word search, synonymy is a problem because a user entering a search word may not retrieve the correct document because the incorrect synonym of the word was used. With polysemy a search word may retrieve spurious documents because of the multiple meanings of the word.

To alleviate these problems, LSI performs a conceptual comparison, which is a statistical analysis of word co-occurrences in documents and identifies repeatable contexts, topics or concepts in which a certain group of words occurs. It then generates a conceptual representation space of all items based on those relations. Within the conceptual representation space, proximity is used to provide a direct measure of the conceptual similarity of any two items represented in that space.

Content Analyst's main Text Analytics capabilities are listed below:

- *Automatic Conceptual Comparison* – This capability is at the heart of Content Analyst functionality. Given any arbitrary block of text (from a single word to an entire book), Content Analyst can automatically map that text block into an appropriate point in a conceptual representation space. It then can provide a direct measure of the conceptual similarity of any two items represented in that space.

- *Automatic Summarisation* – Content Analyst can instantly identify key sentences that most accurately represent the key concepts within a document, and uses those sentences to give users a quick summary of the entire document.

- *Advanced Retrieval* – Operating on the basis of concepts, Content Analyst does not depend on the words that users choose when they formulate queries. As a

result, Content Analyst can retrieve relevant documents even if they contain no words in common with the queries.

- *Automatic Categorisation* – Content Analyst can automatically create relevant categories based on concepts found within documents or other information sources. Content Analyst then automatically assigns incoming documents into these categories.

- *Automatic Taxonomy Generation* – Content Analyst can automatically sort a collection of documents into subsets based on their conceptual content. Sorting is data-driven based solely on the content of the documents, not on any preconceived ideas of how information should be categorised.

Similar to Autonomy the use of statistical modelling means that the system is language independent and linguistic tools are not required.

2.1.4.3. Overview

The main weakness for both Autonomy and Content Analyst is that their information management systems rely too heavily on statistical modelling techniques. They sacrifice language understanding for reduced manual intervention and language independent systems. This means that the systems have no understanding of language, as words and phrases become abstract symbols of meaning deriving a systems understanding through the context of their occurrence.

Take for example the LSI conceptual representation space, where the items within the clusters do not have linguistic categorisation of what they have in common. This is because the dimensions in the space have no interpretable meaning in natural language. Another problem is that the clustering produces an abstract ideal item, which is usually the centre of mass in the space. This item though may not be the most important theme in the space.

Active Navigation uses a hybrid of linguistic and statistical techniques to understand the language in order to build relationships between documents. Fundamental to improving

the Active Navigation navigational solution is this language understanding. The key requirement therefore is the extraction of the key themes from a document.

2.1.5. Summary of Section

This section has provided an overview of Active Navigation. It has outlined its branding in the industrial market place, and discussed the techniques it uses to provide a navigation solution. A couple of competitors were discussed to broaden the perspective of viewing different solutions to the information management problem.

This section has highlighted the importance of extracting key themes from a document and what the extracted material can be used for, in particular the use of themes to provide dynamic linking between documents as well as improving other navigation techniques. The topic of themes and theme extraction is addressed in the next section.

2.2. Themes

This section provides an in-depth overview of themes which is essential for understanding how to build a theme extraction system. This section contains an overview of the different theme theories, an introduction to a traditional theme extraction process which has been shaped by these theme theories, and also the important issue of how to rank extracted themes using different methods. This section concludes with a summary.

2.2.1. Theme Theories

In this section are five proposals on the notion of a theme presented in [Paradis & Berrut, 1996].

- *Dependency* – One can assume that if an item is dependent on another in an expression, then the latter is a theme for that expression. For example, in 'red car', the noun 'car' is considered more important than the adjective 'red'. This idea is the basis of many grammars [van Riemsdijk & Williams, 1986]: head-modifiers, Generalised Phrase Structure Grammar, Lexical Functional Grammar etc. These grammars only apply to head-final languages such as English where the final noun is usually the head of the noun phrase, and any preceding and following words are usually modifiers of the noun.

- *Thema* – In Zemb's statutory analysis [Melby, 1987], the thema corresponds to what the sentence is about, the rhema what is said about it, and the phema how it is said. This idea is closely related to the notion of topic in Information Retrieval. An approximation of the thema, such as using noun phrases, is often thought to be sufficient for Information Retrieval purposes [Berrut & Palmer, 1986].

- *Topic / Focus* – Topic Focus Articulation looks at the progression of themes across sentences [Hajicova & Sgall, 1984]. The idea is that a topic or a focus that is repeated in a subsequent sentence can be considered as a theme. This approach is similar to term frequency in adjacent sentences, so suffers from a loss of information as only the word count is used, and all other information becomes irrelevant.

- *Given / New* – Given or old information is supposedly known to the user, as opposed to new information, and as such can be considered to belong to the theme.

- *Intentionality* – The intention corresponds to the idea(s) the author had in mind when they wrote the document; Grosz and Sidner suggest in [Grosz & Sidner, 1986] that they can be assimilated to topics. Intentions are also reflected in linguistics theories of meaning [Grice, 1971]. Section or document headers can be considered quite informative on the author's intentions in explanatory texts. It is suggested in [Hirschberg & Litman, 1993] that some cue phrases are a direct indication of the topics. Meta-discourse expressions such as 'In this paper we describe...' are quite common in scientific literature, and it is hypothesised that people use those expressions for determining the themes.

The following two points offer ideas on how to derive themes from logical structure and domain knowledge [Paradis & Berrut, 1996].

- *Logical Structure* – The logical structure of the text can help identify themes, as it reflects the structuring of the topics themselves [Hearst & Plaunt, 1993]. This idea was already used in Information Retrieval in the mid-eighties [Chiaramella et al., 1986]. In other words the importance of a theme is inferred from the logical structure of the document.

- *Domain Knowledge* – Given a theme one can deduce other themes using domain knowledge and relationships such as synonymy. It is important to understand the semantics of the theme in order to form deductions. In order to do this the domain-specific knowledge, both the context and the vocabulary, must be understood.

2.2.2. Theme Extraction

2.2.2.1. Introduction

Theme extraction is the process of automatically identifying and extracting pertinent information from unstructured information.

Traditional theme extraction uses statistical modelling to process documents and extract themes mainly based on noun phrases and to find relationships between them.

Some statistical methods are even more simplistic and treat theme extraction as a word frequency approach. With this method all sense of word order and context is discounted as the only focus is the occurrences of words. A word frequency approach can sometimes derive some correct themes for a document, but generally it misses the key themes as it does not take context into account. The loss of information in this method is prominent, and affects any post extraction techniques.

It is hypothesised that Natural Language Processing techniques can help to provide better representation of documents for information navigation purposes than any simple word based methods commonly used in statistical modelling [Strzalkowski et al., 2000].

This is based on the premise that linguistic processing can uncover certain critical syntactic and semantic aspects of documents that simple word counting cannot do, and this should lead to a more accurate representation.

2.2.2.2. Traditional Method

Theme extraction requires understanding of Information Extraction (IE) and Information Retrieval (IR). It is important to distinguish between the two and identify the position that theme extraction covers.

IE is the extraction of facts using templates to populate a database. IR is used in the retrieval of documents. Theme extraction covers parts of IE and IR as it extracts information from texts that can then be used for document retrieval. The process of extracting information from texts in theme extraction can be viewed as being similar to Named Entity recognition.

The theme extraction process requires a substantial quantity of work to build up an architecture that is capable of extracting themes from unstructured information. The main components used in a theme extraction architecture include a combination of linguistic (knowledge-based) and statistical tools. This can be viewed as a linear path consisting of the following steps:

- *Tokenisation* – The text content of a document must be delimited so that latter components can process the token stream appropriately.

- *Lexical Analysis* – Lexical analysis usually has two parts. The first is a lexicon of syntactic and semantic information about words and idiomatic phrases. Secondly a collection of morphological rules for recognising regularly inflected forms of words, for recognising words derived from other words by the addition of prefixes and suffixes, for recognising words formed as compounds of known words (e.g. 'shoelace'), and for guessing syntactic and semantic information about unknown words [Woods, 1997].

- *Stemming* – This is the normalisation of words, where the root form of a word is extracted. Although stemming is useful in finding other words that have small derivations, it losses inflectional information from the word and can be inaccurate because they can be too aggressive or too passive in nature.

- *Word-Class Assignment and Word Sense Disambiguation* – Determines which word-classes and word sense (e.g. 'suit' can be either about clothing or with legal cases) apply, given the particular context in which the word occurs [Cardie, 1993].

- *Phrase Analysis* – This is the formation of syntactic structures traditionally using the word-class information derived from the token stream. A parser and grammar are vital for recognising the structures [Woods, 1997]. Usually noun phrases are extracted from the syntactic structures to represent candidate themes.

- *Pruning of Modifiers* – Pruning is a method that loses information. Some losses are acceptable however. For example the determiner word-class can often be stripped from the beginning of noun phrases whilst losing only losing minimal information, but when adjective and adverb word-classes are pruned the essence and meaning of the theme are lost and this is unacceptable. E.g. in the theme phrase 'the big red car' the determiner 'the' can be pruned but 'big' and 'red' are preserved. This pruning of noun phrase modifiers uses a stop-word list which is a list of function words that do not add content to a theme.

- *Ranking Themes* – Themes that are not representational of the document are masked. Some linguistic methods for filtering out irrelevant themes for a user are based on either a thesaurus or a lexicon [Nakata et al., 1998]. This is particularly relevant for domain-specific documents or users with a specific interest. Some statistical methods are scoring strategies or the use of clustering techniques where the centre of mass and items with a close proximity to it are the ideal themes to represent the document.

The main problem with current theme extraction systems is that components in the architecture make decisions which lead to only a single output being presented to the next component. This 'best-guess' approach means that information is lost, which critically means that the correct decision is lost or the wrong decision is passed onto the next component. This means that errors are likely to propagate through the system.

To accomplish theme extraction more accurately requires retention of information, and so adhere to a lossless ethos. Otherwise as soon as information is lost, there is no way of getting it back.

2.2.3. Ranking Themes

Representing the relative importance of a theme requires ranking and ordering of it against others. The ranking of themes requires a weighted calculation of a number of measurable variables available from the document to determine a themes importance. Some of the most important ones are explained in the next sections. They can be categorised by the scope that they work in. These different scopes are document, sentence, phrase, and word. It is a combination of these variables that decide key themes, not just on one particular one.

2.2.3.1. Document Scope

- *Position in Document* – The position of a theme in the document structure can be an indicator of its importance. Major headings and sub-headings usually function as concise summaries of their related content bodies, and as such can be interpreted as key themes. However, some of these do not represent the document in any way. Document structure alone is not enough reason to extract

a key theme; more contextual information is needed to determine how representative the theme is of the document.

- *Frequency of Occurrence* – The frequency of occurrence of a theme and of its synonyms, anaphoric references and derivations, usually indicates it as a key theme. Critically, sometimes the more subtle, lower occurrence themes can also be considered to be important.

- *Aesthetic Cues* – When documents are written, it is human perception that allows the addition of aesthetic cues to enhance the intentions of an author's original document. Factors such as colour, font, size and style can help determine the importance of particular parts in a document. They can signify parts of the document that are of interest to the reader, or parts that are more important than other parts, but it does not mean that the part has to be a theme.

2.2.3.2. Sentence Scope

- *Syntactic Function* – The word order in a sentence gives an indication of importance. It determines the syntactic function of different parts of a sentence and aids in identifying potential themes such as the subject and objects of a sentence.

- *Dependency* – Word dependencies can often identify the potential theme amongst a group of words. For example the term 'big red car', the words 'big' and 'red' are dependent on the word 'car' and therefore 'car' has more importance.

2.2.3.3. Phrase Scope

- *Explicitness of Context* – Longer phrases contain more context than single words. This context is a primary discriminator of importance. In general longer themes are preferred to shorter ones as they are more specific representations of a document. For example 'Prime Minister Tony Blair' is preferred to 'Blair'.

  As a longer theme stores more context there is a possibility that it can be expanded into multiple shorter themes. These shorter themes can be unique

themes and the conceptual similarity between the short themes and the longer theme is close.

- *Function Words* – Function words (aka closed-class words) can join smaller parts together to form a theme. The affect of this is that the theme contains more information and it can also allow the deriving of themes that do not exist in the text. Understanding the syntax that each function word imposes is essential for this operation. For example the theme 'safe handling and use of concrete blocks' can be interpreted as two themes 'safe handling of concrete blocks' and 'use of concrete blocks', even though they are not explicit given the linearity of the text.

- *Modifiers* – A theme can have any number of modifiers. The general hypothesis is that more modifiers are better, but up to an unspecified limit. The key decision is to stop adding modifiers that do not add value to the theme.

  One solution for determining the number of modifiers is to use a weight for each modifier. This assigns a weight for each modifier based on its word-class. The product of the modifiers defines how long the modifier chain is. The chain starts at the theme head and modifiers are only allowed if the product remains above a threshold.

  It is also hypothesised that the proximity of a modifier to the theme head dictates its importance, so that close or adjacent modifiers are more important than ones that are further away. E.g. in the phrase 'nice blue metallic car', the modifiers 'nice' and 'blue' have less affect on the theme head 'car' than the modifier 'metallic'.

2.2.3.4. Word Scope

- *Word-Class Assignment* – A word-class is a syntactic tag that identifies the type of function that a word serves. Often themes are nouns, but in certain cases they can be of other types.

- *Capitalisation, Proper Nouns, Acronyms and Abbreviations* – Capitalisation of a word usually indicates a proper noun, acronym or abbreviation, which are often

candidate themes. However not all of these are considered a theme. The major problem is ambiguity. For example some words are difficult to distinguish without extra context. E.g. 'May' can be a name, month, or modal auxiliary verb.

- *Common Nouns* – Common nouns are often candidate themes.

- *Times, Dates and Numbers* – The presence of times, dates, and numbers usually indicates a presence of a theme. These entities often directly modify a theme, but are sometimes themes on their own right. E.g. '9/11'.

2.2.4. Summary of Section

This section has provided an in-depth overview of theme theories, a typical theme extraction process, and methods to rank extracted themes.

The theme theories have defined different strategies to identify themes in text. The author's theme extraction system will aim to be similar to traditional Information Retrieval systems and to this end it will use an approximation of thema theory, which is to extract noun phrases, and an approximation of topic/focus theory so that term frequencies can be used. The system will not be using the dependency, given/new or intentionality theories as they require too much semantic knowledge. It should be noted that most of these theme theories assume at least one theme per sentence and there is no such requirement on the author's system.

This section has also highlighted the various ways that themes can be interpreted from text and how an automated system can use linguistic methods to extract these themes and output the most pertinent ones.

The knowledge from this section is used in designing and implementing a theme extraction system for evaluation with Active Navigation.

3. Background of Natural Language Processing

Natural Language Processing (NLP) is the application of Artificial Intelligence techniques to answer questions about text written in a human language. The question being asked for this research is how to extract themes from the text.

Understanding of theme extraction first requires an in-depth look into the background history of NLP, in particular the constituency and dependency language models on which all Information Extraction systems are based.

This chapter provides a detailed discussion into the two polarised language models of constituency and dependency, and then parsers are discussed for the language models as both use grammars to form syntactic representations.

Some benchmark systems for both language models are compared and contrasted; this shows that the type of model strongly influences the architecture of a system. A summary concludes the chapter.

3.1. Constituency Language Model

A constituency grammar is the basis of formal language theory as studied by computer scientists [Covington, 2000]. It is a formal interpretation of a language and is expressed by a set of grammar rules which provide the coverage of the language.

Constituents are syntactic structures that bracket together parts of a sentence, based on the grammar rules. The bracketing can be at the lowest level, such as phrase bracketing or at higher levels, such as clause and sentence bracketing. Constituents are not allowed to overlap and in this vain the constituents form a hierarchical tree structure, often referred to as the constituency tree (see Figure 3-1). Each level of attachment in the constituency tree represents larger constituents.

At the bottom of the tree are the terminal nodes, these are the smallest lexical element and are often individual words. Each word is given a classification based on its word class. For example the word 'an' is tagged as a determiner (D) and the word 'grammar' as

a noun (N). The non-terminal nodes define the constituents. For example the preposition phrase (PP) consists of a preposition (P) and a noun phrase (NP). The root of a constituency tree defines a valid sentence.

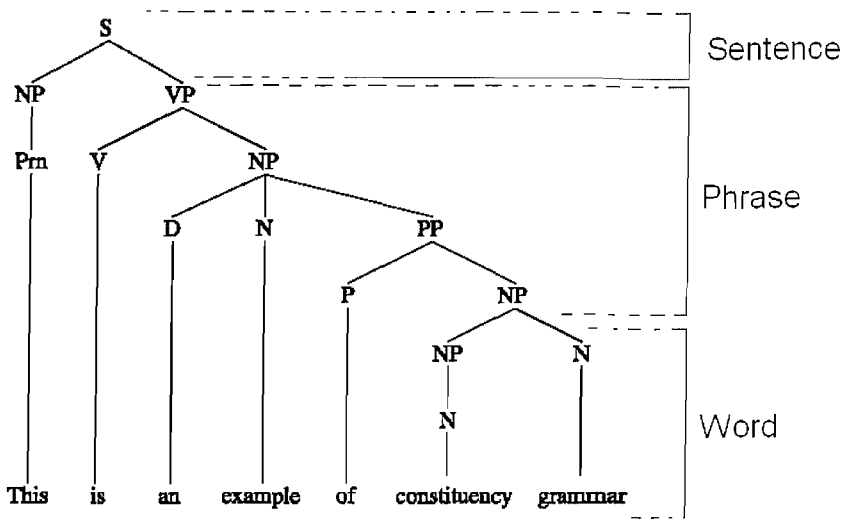

Figure 3-1 – A constituency tree [Covington, 2000]

Constituency grammars are popular as language models largely because they are easily modelled by the context-free phrase-structure grammars [Covington, 1990]. The context-free grammars (CFGs) come from a branch of mathematics rather than of natural language. Given the nature of CFGs it is obvious that they do not have enough expressive power for natural language as they require unambiguous grammars. Natural language though is extremely ambiguous.

Multiple tree interpretations occur in constituency grammars due to attachment ambiguities and word class ambiguities. It is left up to the grammar rules to resolve these ambiguities.

Although constituency grammars have been highly successful with English language, it has trouble with variable word orders for two reasons. First the order of the constituents is variable, second, and more seriously, variable word order often results in discontinuous constituents. Although a constituency grammar may be rich enough to handle free word orders if it is combined with a transformational grammar.

3.2. Dependency Language Model

The latest syntactic theories put less and less emphasis on trees as a representation of structure. The emphasis is shifting toward the grammatical relations (aka dependency relationships) that link the head of the phrase to the other constituents of the phrase [Covington, 1990].

The dependency grammar structure is a directed acyclic graph (see Figure 3-2). With nodes representing lexical elements and edges representing dependency relationships. Normally there is a requirement that the graph is connected and acyclic, which means that it is rooted graph with the root node representing the head of the sentence. In constituency grammar the root is the entire sentence, but in dependency grammar the root is the main verb, this is because they have sub-categorical arguments (subject, object etc.) as their dependents.

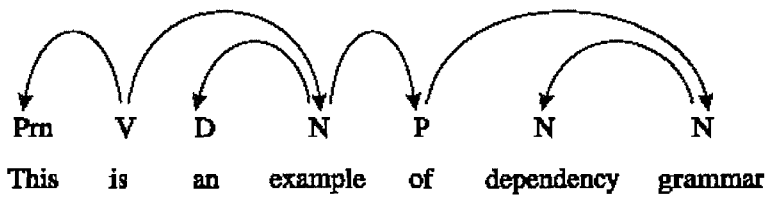

Figure 3-2 – A dependency relationship graph [Covington, 2000]

In general, the dependent is the modifier, object or complement, it can either precede (pre-dependent) or follow its head (post-dependent). A word is independent if it is not a dependent of any other word. The head plays the larger role in determining the behaviour of the pair, and therefore the notation for the edges usually point from the head to the dependent.

The dependent presupposes the presence of the head. I.e. adjectives depend on nouns, not vice versa. Whereas the head does not require the presence of the dependent.

The goal of a dependency grammar is to describe syntactic analysis of sentences using dependency relations that show the head-dependent relations between words. This is an improvement on constituency grammars as it allows explicit syntactic disambiguation of a

head and its dependents (see Figure 3-3), and the dependency relationships are close to the semantic relationships needed for the next stage of interpretation. A dependency relationship can in theory relate one word to another across the entire sentence; this is an improvement on the constituency grammars, because most of the phrase structures are limited to local context.

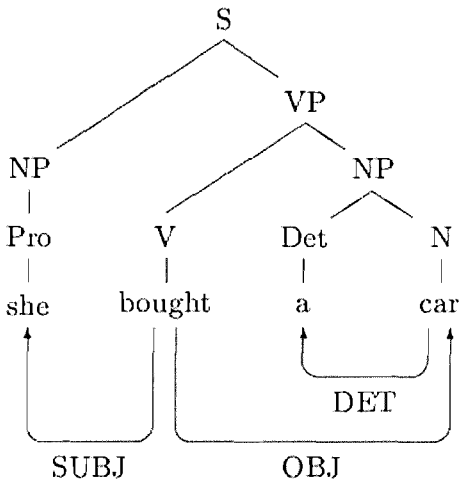

Figure 3-3 – Comparing a constituency tree with a dependency graph [Nivre, 2002] and [Nivre & Nilsson, 2003]

In a dependency graph, constituents still exist but they are derived rather than a fundamental concept; for example a phrase consists of any headword together with all the words that depend on it. Following the modifier chains of a headword in both directions, to the most distant modifiers, yields the phrase headed by the word. The label for the phrase can be generated from the word class of its headword. Therefore the common formal property of dependency structures, as compared to the more common syntactic representations based on constituency, the lack of phrasal nodes becomes a grey area.

There can be a number of parameters that can vary between different dependency grammars. For example, lexical elements can be assumed to represent words, strings-of-words or even parts-of-words; and dependency relationships can be labelled with syntactic functions, semantic roles or not at all. Dependency grammars also vary in the model that they use, but most incorporate these principal features:

- *Uniqueness* – Every element of the dependency graph has a unique head. The verb serves as the head of a clause, and the top element of the sentence is thus the main verb of the main clause.

- *Projectivity* (*aka Adjacency*) – A head and its dependent can only be separated by other dependents of the same headword. More formally, A is adjacent to B provided that every word between A and B is a subordinate to B. In a linear dependency grammar this amounts to the requirement that there are no crossing edges. There is a strong correlation between projectivity and linearity, in that linear dependency grammars tend to assume projectivity, whereas non-projective grammars typically use non-linear dependency graphs.

- *Valency* – This governs which arguments are expected, it describes the number, and type of modifier a dependency can have. Usually complements (obligatory) and adjuncts (optional) are distinguished.

- *Multiple Parse Trees* – As with constituent grammars, multiple parse trees can occur, this is usually when a word has several syntactic functions with respective dependency relatives. This forms a dependency forest. Global pruning is required to extract the dependency relationships that form consistent trees.

3.3. Current Parsers

Both language models depend on a grammar to build syntactic structures. Independent of the choice of language model, a grammar requires a parser to process the information. A parser is a program that determines the structure of a sentence. This definition of parsing is taken from [Aho et al., 1986], "Parsing is the process of determining if a string of tokens can be generated from the grammar".

Constituency and dependency language models both have many parsing strategies. The next couple of sections aim to provide an overall guide into the state-of-the-art parsers available for both language models.

3.3.1. Constituency Parsers

Constituency parsers frequently adopt a bottom-up approach to building constituents, and therefore rules need to be defined to encapsulate the scope of each and every constituent, as well as an efficient rule selection process in ambiguous cases. In reality this is a tough task as a structured approach cannot deal with poorly formed items such as sentences.

3.3.1.1. Head Parsing

A major branch of constituency parsing is head parsing where the head of each phrase is passed up the tree. Head parsing is not just a parsing technique but also helps to add extra information to the parse tree, in this case showing the heads at each non-terminal node (see Figure 3-4).

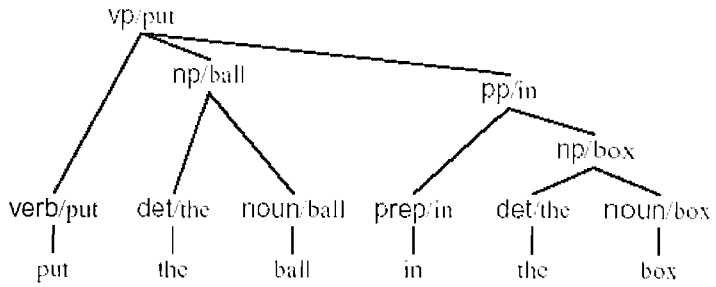

Figure 3-4 – A constituency tree showing head information [Charniak, 2001]

Charniak's immediate head parser [Charniak, 2001] is a statistical left-to-right parser using a trigram model. For each sentence position in left-to-right parsing the probability of the next word is computed based upon the previous two words. These probabilities cannot be modified. The immediate head parsing model assigns a probability to a parse T by a top-down process of considering each constituent C in T.

All of the properties of the immediate descendents of a constituent C are assigned probabilities that are conditioned on the lexical head of C. For example in Figure 3-4, the probability that the VP expands into 'Verb NP PP' is conditioned on the head of the VP, as are the choices of the sub-heads under the VP, such as 'ball' (the head of the NP) and 'in' (the head of the PP).

A head parser has the advantage of working from heads out through islands of certainty before dealing with less certain regions of a parse. Head parsers combine top-down and head-outward bottom-up parsing, which gives them the distinctive feature of sometimes working over discontinuous regions around heads rather than left-to-right.

The problem with head parsing is the cost in holding unattached constituents in head-final languages such as Japanese, Korean and German.

3.3.1.2. Shift-Reduce Left-to-Right Parsing

Most constituency language models adopt a left-to-right parse with a k-token lookahead. The reasoning behind this is that most languages are read from left-to-right and predictions are automatically made by the reader of what words occur next.

These parsers have a stack and as input the first k-tokens of the input are the lookahead [Appel, 1998]. Based on the contents of the stack and the lookahead the parser performs two kinds of action, either 'shift', which moves the first input token to the top of the stack or 'reduce', when it chooses a grammar rule X → ABC; pop C, B and A from the top of the stack; push X onto the stack. Initially the stack is empty and the parser is at the beginning of the input. The action of shifting the end-of-file marker is called 'accepting' and causes the parser to stop successfully.

The problem with left-to-right parsing is the possibility of obsolete rules, and more importantly errors cannot be corrected later if an incorrect shift or reduction is made.

Ambiguities arise when more than one operation is allowed. A shift-reduce parser encounters ambiguity in three possible places [Mellish, 2004]:

- *Shift / Shift Ambiguity* – arises when a word is lexically ambiguous.
- *Reduce / Reduce Ambiguity* – arises when more than one sequence of categories at the top of the stack can be reduced (or a sequence can be reduced in more than one way).
- *Shift / Reduce Ambiguity* – arises when both a shift and reduce are possible at some point.

3.3.1.3. Problems of Deterministic Parsers

Head parsing and shift-reduce left-to-right parsing are both forms of deterministic parsing. Deterministic parsers have unrecoverable problems when faced with issues such as interpreting an ambiguous grammar, handling sentence context, and performing robust parsing of input sentences.

Natural language is not deterministic in nature. It is ambiguous and complex. A grammar that aims to model natural language needs to provide coverage for ambiguous interpretations of arbitrary input. Unfortunately a deterministic parser cannot manage an ambiguous grammar mainly for two reasons.

The first is that it uses an unambiguous grammar to make parsing of input simpler. This forms a distorted grammar as it attempts to fit a grammar to a deterministic parser. This

misses the point of modelling natural language as it is the grammar that is paramount, not the parsing technique. Trying to make a grammar unambiguous loses the structure of a language.

Secondly deterministic parsers only provide a single syntactic representation as output, which means that at any ambiguous point in an input, disambiguating rules are used to decide which path to take. All other possibilities are discarded. These rules are simplistic in nature. Such as always choosing shift over reduce when conflicts arise; choosing first matching rule from a set of possible matching rules; or making a decision only using a single-token lookahead which lacks context. The output is therefore dependent on the disambiguating rules and not actually on the context of the input, this means that the output can be incorrect and potentially correct representations are lost.

The author assumed that a natural language can never be fully derivable by a set of grammar rules. Therefore the grammar is always incomplete. This means that some input cannot be parsed as there is no suitable grammar rules to consume the input. A deterministic parser therefore cannot accept the input. A more desirable parser must be more robust and be able to manage complex input in an appropriate way.

In summary a deterministic parser can only produce one parse per input string, and it is unable to produce ambiguous multiple parses which means that information is lost. The solution is to use non-deterministic parsers.

3.3.1.4. Non-Deterministic Parsers

There are many variations of non-deterministic parsers, the most important of these are Generalised Left-Right (GLR), Tomita, and Earley parsers [Aycock & Horspool, 1999, 2001, 2002] [Aycock et al., 2001] [van den Brand et al., 2002] [Horspool & McLean, 1996] [Inui et al., 1997] [Lavie & Tomita, 1993] [Marino et al., 1987] [McPeak & Necula, 2004] [Numazaki & Tanaka, 1990] [Thorup, 1994] [Tomita, 1990] and [Wagner & Graham, 1997].

In essence they all perform the same thing which is to simulate parallel execution of multiple copies of a left-right parse, and so simulating non-determinism and effectively taking all actions rather than deciding on a single action.

A non-deterministic parser operates on the same principles as left-right parsers, but they can be used with all arbitrary context-free grammars, including ambiguous grammar. The parser acts as a normal left-right parser until a conflict forms, it then clones new copies of itself to track each of the conflicting actions simultaneously. Some copies of the parse subsequently reach a state where parsing cannot proceed and these copies of the parsers simply terminate execution.

A non-deterministic parser does not place any restrictions on the grammar, which means that the parser instead fits around the grammar and making the modelling of human language more accurate. It also means that information is not lost, and so all syntactic representations of the input are retained.

3.3.2. Dependency Parsers

Dependency parsing is by accepting and attaching words one at a time rather than waiting for complete phrases. This is claimed to be closer to human parsing [Abney, 1989] as no presumptions are made in advance. This overcomes the psycholinguistic objection to shift-reduce parsing. As the parser's job is only to connect existing nodes, not to postulate new ones (unlike constituency grammars), the task of parsing is in some sense more straight forward.

3.3.2.1. Statistical Dependency Parsing

The family tree of dependency parsers has many branches, most of the work is based on 'statistical parsers' [Berger & Printz, 1998a, 1998b] [Carroll & Bricoe, 2001] [Carroll & Charniak, 1992] [Chelba et al., 1997] [Infante-Lopez et al., 2002] [Nivre, 2002, 2003] and [Paskin, 2001]. In statistical dependency parsing the disambiguation is performed using decreasing probabilities and constraints, the main choice to be made is the learning algorithm to be used.

The probabilities are extracted by running a pass over the training data to extract the potential information to be used and to use these statistics to form the basis of the model. The training data tends to be either unmarked text, or marked up with word class information using a constituency tool called a part-of-speech tagger. The training data is never marked up with dependency information.

The major weaknesses of these approaches are the lack of linguistic information, lack of syntactic labelling of the dependency relationship and they are limited to projective grammars. Being of a statistical nature they also suffer from sensitivity to sparse data and the difficulty of setting heuristic measures.

3.3.2.2. Deterministic Dependency Parsing

'Deterministic parsers' advocated by [Arnola, 1998] and [Nivre & Nilsson, 2003] tend to build a syntactic analysis based on efficient disambiguation, high accuracy and robustness. These parsers though can suffer from a lack of labelled links, and the robustness feature means that a rooted tree is not required, so there is no projectivity constraint and a complete dependency structure does not need to be found. With deterministic parsers, once a link had been added it cannot be removed and therefore blocks the addition of other possible edges, even if the link is later found to be in error. This also means that local syntactic relationships tend to be formed.

Deterministic dependency parsing can be viewed as an interesting compromise between so-called deep and shallow processing. It is a kind of deep process in that the goal is to build a complete syntactic analysis for the input string, not just identify basic constituents as in partial parsing, but it resembles shallow processing in being robust, efficient and deterministic. For certain applications of dependency parsing, such as the detection of dependency relations for Information Retrieval in unrestricted text, the level of precision, efficiency, and robustness obtained can be sufficient to justify the use of a deterministic parser.

Deterministic dependency parsers have two major sub-branches, those of 'block-based parsers' [Krymolowski & Dagan, 2002] [Kübler & Hinrichs, 2001] [Zhou, 2002] and 'finite-state parsers' [Ciravegna & Lavelli, 1999] [Elworthy, 1999] and [Oflazer, 1999].

Block-based dependency parsers use a constituency-based partial parser plus dependency parsing on the resulting chunks (a chunk is usually a syntactic structure such as a phrase). Chunks are usually non-recursive syntactic structures which form larger recursive structures. Such larger structures are not only desirable for a deeper syntactic analyses but they also constitute a necessary prerequisite for assigning function-argument structure that aids robustness by dealing with unrestricted texts.

Traditional dependency grammar builds dependency relations between any of two specific words, which require a large word-based grammar to be produced, which is a demanding task, and fully annotated treebanks (repositories of manually annotated text) and specific texts in unique domains, or languages other than English are not readily available. Block-based parsing uses constituency parsing to compensate for the lack of knowledge of word-based dependency parsing, and reduces the scope of dependency relations between dependents and heads of chunks rather than just individual words.

The partial parser produces a structure in which some constituents remain unattached or partially annotated, in keeping with the partial parsing strategy to factor out recursion and to resolve only unambiguous attachments. Parsing proceeds by growing islands of certainty into larger and larger phrases. The major advantage of partial parsing is that ambiguities are never passed on to higher levels, as the parser is generally aiming to annotate only partial, reliably discoverable tree structures, the result is containment of ambiguity. But some phrases are never resolved, and so some constituents are never attached.

The finite-state dependency parsers use different levels (or cascades) to produce a dependency structure output. Each level contains parse rules for one or more phrase types, and passes on completed phrases and unparsed elements to the next level. The basis of finite-state parsing is computational efficiency, lower time complexity, and easy integration of levels of processing (modularisation).

The parser attempts to form a phrase from the longest prefix of the input which matches a rule in the grammar. If it succeeds, it continues with the remainder of the input, if not, the word remains unparsed and the parser resumes with the following word. There is no recursion so phrases never contain same-level or high-level phrases. The spans of input elements are reduced to single elements in each finite transduction, as in traditional parsing.

Finally dependency links are constructed by means of transducers which introduce 'channels' representing potential locations for dependency links. The actual links are then constructed for the lexical items and morphemes that are specified to have suitable potential connections and constraints such as projectivity are respected. The difficulty

with this approach is that the grammar must be deterministic (I.e. no conflicting left-to-right longest match rules).

3.3.2.3. Discontinuous and Functional Dependency Parsing

A couple of other influential dependency parsers are Covington's 'discontinuous dependency parser' [Covington, 1990, 1994, 2000] and Järvinen and Tapanainen's 'functional dependency parser' [Järvinen & Tapanainen, 1998] and [Tapanainen & Järvinen, 1997].

The unique selling point of the discontinuous dependency parser is that it can model both fixed and variable word orders providing unlimited word order freedom by using a non-deterministic algorithm. Discontinuous dependency parsing treats free word order as the simplest case, and treats restrictions on word order as additional constraints. Despite being able to handle unlimited word-order freedom, the parser has a psychologically realistic preference for near attachment, as it tries near attachments first. To adopt a fixed word order, the parser needs to be able to specify that a phrase must be continuous, then specify the order of the head relative to each of its dependents and finally specify the mutual order of the dependents of a head.

The functional dependency parser is an extension to the original 'constraint grammar parser of English'[5] by adding a dependency parser layer. The reason for this is that the syntactic tagset of the constraint grammar provided an under-specific dependency description. The words can be defined as heads but the parent cannot be indicated, whereas with the functional dependency parser dependencies are made clear by declaring the heads and dependents. As the original constraint grammar was shallow, differing tags may have received the same type of label therefore a considerable amount of syntactic ambiguity cannot have been resolved reliably.

3.3.2.4. Constraint Satisfaction Dependency Parsing

Finally there are 'constraint satisfaction parsers' [Blache, 2000] and [Menzel & Schröder, 1990]. In contrast to the traditional view on parsing as a constructive process, which

[5] http://www.lingsoft.fi/doc/engcg/intro

builds new tree structures from elementary building blocks and intermediate results, eliminative approaches organise structural analysis as a candidate elimination procedure, removing unsuitable interpretations from a maximum set of possible ones. Hence parsing is constructed as a strictly monotonic process of ambiguity reduction.

In this approach, constraints are not directly expressed over linguistic objects, but over the relations. This entails a passive use of constraints in the sense that the parsing process consists in building first these relations, and then verifies that they satisfy the constraints. Practically, this has an impact on the scope of the constraint which can only be local and applied to a given structure. Being reductionist in nature it allows arbitrary categories (not just the head verb) to serve as the top node of the dependency graph, of course these configurations need to be penalised appropriately but it does give the grammar an option if no alternative interpretations remain.

Constraint parsers can use weighted violable constraints. Violable constraints allow marking up of some analyses as better as others, giving the flexibility to provide analyses for even highly ungrammatical input; this improves the robustness of the parser. Flexible constraints better model human parsing as it is robust towards grammatical errors in the input, and can disambiguate between alternate analyses by comparing their plausibility and picking the best. A complete disambiguation is achieved provided that enough preferential knowledge is encoded by the constraints.

3.3.3. The Rationale for Parsing

The reason for parsing is the formation of syntactic structures from which to extract themes. This is the author's chosen method to determine the structure and to identify the important information of a sentence. An alternative option for identifying important information would be to use the Minimum Description Length (MDL) [Rissanen, 1978] and [Grünwald, 2004].

MDL is a method for inductive inference to infer general laws and principles from particular instances. The first central MDL idea is based on the insight that every regularity in data may be used to compress that data. The more regularities there are, the more the data can be compressed. The second central idea is that learning can be

equated with finding regularities in data. In other words, the more we are able to compress the data, the more we have learned about the regularities underlying the data.

In respect to natural language, MDL can be used to substantially compress text, as long as it is syntactically mostly correct. By first describing a grammar for that language, and then describing a text T with the help of that grammar, T can be described using much less bits than are needed without the assumption that word order is constrained [Grünwald, 2004].

As MDL performs statistical modelling over text, it cannot capture linguistic information that can identify themes and it cannot identify key themes that occur infrequently. By parsing the text the actual linguistic structures are generated and the importance of these structures are analysed to extract the key themes.

3.4. Example Systems

This section is an overview of some of the systems used in Natural Language Processing. The first four are constituency based whilst the final system is similar to a dependency model. The General Architecture for Text Engineering system is a pipeline for Information Extraction. It is a solid starting point for understanding the types of components that are required in a constituency pipeline and their interactions with each other. The Porter stemmer is a suffix stripping system that can be used as a morphology component in a constituency pipeline. It is a component that looks trivial, but in fact deals with a complex problem. The Brill part-of-speech tagger deals with a fundamental part of a constituency model, it assigns word-class to words. It is arguably the most important component in the constituency pipeline, as word-class information is required in higher levels in order to build up larger constituents and for ambiguity resolution. The Treebank II bracket parser shows how powerful a constituent grammar can be given a rich expressive tagset that can deal with sentence level structures, the only problem is that it is manually annotated and so cannot be used in automatic Information Extraction at present. Finally the link grammar parser based on link grammar an original theory of English syntax. It is very similar to the dependency model in that it is builds up relationships between words, but it is far more expressive than any dependency model.

3.4.1. General Architecture for Text Engineering (GATE)

GATE[6] comprises an architecture, framework and development environment, and has been in development since 1995 in the Sheffield Natural Language Processing group [Maynard et al., 2000] and [Cunningham et al., 1999, 2002a, 2002b]. The system has been used for many language processing projects, in particular for Information Extraction in many languages.

GATE version 2 is distributed with a shallow Information Extraction system called ANNIE[7] (A Nearly New Information Extraction system) [Cunningham et al., 2002b] it is combined with LaSIE components (from the original first version of GATE) when doing complex extraction tasks (see Figure 3-5). ANNIE uses the JAPE (Java Annotation Patterns

[6] http://www.gate.ac.uk

[7] http://www.gate.ac.uk/sale/tao

Engine) language developed by Hamish Cunningham and Valentin Tablin. JAPE provides finite state transduction over annotations based on regular expressions.

A JAPE grammar consists of a set of phases, each of which consists of a set of pattern / action rules. The phases run sequentially and constitute a cascade of finite state transducers over annotations. The left hand side (LHS) of the rules consist of an annotation pattern that can contain regular expression operators. The right hand side (RHS) consists of annotation manipulation statements. Annotations matched on the LHS of a rule fire the action on the RHS.

The GATE approach to rule matching requires explicit pattern details, and it is possible that a lot of information is not captured using regular expression matching.

The system requires three main processing resources: a tokeniser, a gazetteer and a finite-state transduction grammar [Maynard et al., 2001].

A 'tokeniser' splits text into simple tokens. The aim is to limit the work of the tokeniser to maximise efficiency, and enable greater flexibility by placing the burden of analysis on the grammars. This means that the tokeniser does not need to be modified for different applications or text types.

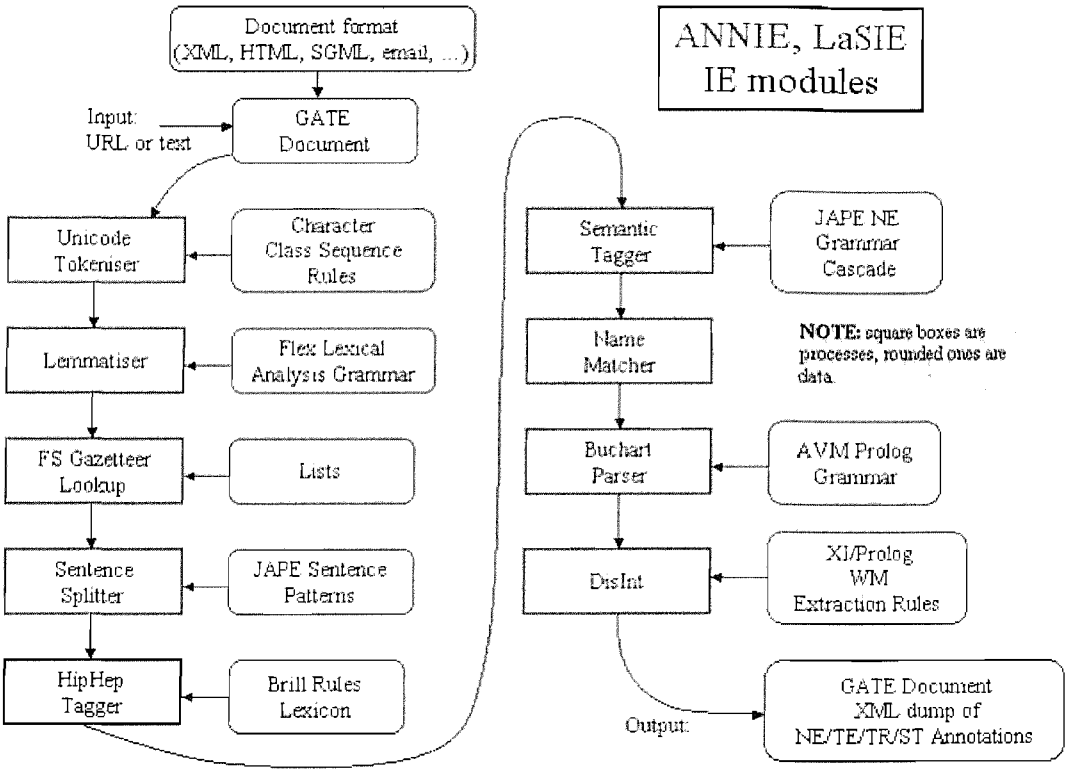

Figure 3-5 – A full Information Extraction pipeline based on a LaSIE backend with ANNIE shallow analysis

A 'gazetteer' consists of lists of entities and names of useful indicators in order to help disambiguate pseudo sentence endings and to extract domain specific information.

The 'grammar' consists of hand-crafted rules describing patterns to match and annotations to be created as a result. Patterns can be specified by describing a specific text string, or annotations previously attached to tokens. Rule prioritisation, if activated, prevents multiple assignments of annotations to the same text string.

3.4.2. Porter Stemmer

This stemming algorithm[8] was designed by Martin Porter [Porter, 1980]. It is a process for removing the common morphological and inflexional endings from words in English. Its main use is term normalisation for Information Retrieval Systems (IRS).

The idea is that a document is represented by terms, terms with a common stem usually have a similar meaning, and the performance of an IRS can be improved if the term groups are conflated into a single term. This is achieved by removal of various suffixes to leave a single root term. In addition, this reduces the total number of terms in the IRS and hence reduces the size and complexity of the data in the system. Porter tested his stemmer on a vocabulary of 10k words the resulting vocabulary of stems contained 6,370 distinct entries. Thus the suffix stripping process reduced the size of the vocabulary by about one third.

The reduction rules appear as (Condition) S1 → S2, where if the word ends with S1 and the stem before S1 satisfies the given condition then S1 is replaced by S2. Only one rule is applied, the one with the longest matching suffix S1 of the given word, this is true whether the rule succeeds or fails (I.e. whether or not S2 replaces S1).

The Porter stemmer iteratively removes suffixes in linear steps. The idea is that the suffixes in the English language (approximately 1,200) are mostly made up of a combination of smaller and simpler suffixes, thus stemming is approached in steps. The stemmer has a five-step process and there are 1 to 21 rules per step. Within each step only the first valid rule fires and it rewrites the suffix.

If a rule is not accepted (fails its condition) then the next rule in the step is tested, until either a rule from that step fires and control passes to the next step, or there are no more rules in that step and control moves to the next step.

Stemmers using very dumb rules work well for English, but unavoidably some errors get generated by the stemmer, for example the stemmer can over-generalise: 'organisation' → 'organ', 'policy' → 'police', 'army' → 'arm'.

---

[8] http://www.tartarus.org/~martin/PorterStemmer

Given two terms, there is some variation in opinion as to whether they are conflated, for example if 'sand' and 'sander' get conflated so are 'wand' and 'wander', the error here is that the '–er' of 'wander' has been treated as a suffix when in fact it is part of the stem. Equally a suffix can completely alter the meaning of the word, in which case removal is unhelpful.

Although the stemmer can make some bad conflations, using corpus-based analysis of word variants indicates which words do not belong together. For example using an Expected Mutual Information Measure (the co-occurrence of words in context) can conclude that the words are probably not related. This information can be used to create two disjoint classes rather than one incorrect stem.

3.4.3. Brill Part-Of-Speech Tagger (POS Tagger)

This POS Tagger[9] was designed by Eric Brill [Brill, 1994]. It is a transformation-based error-driven learning module (see Figure 3-6). It is supervised learning using a manually annotated corpus. It learns language structure by a sequence of transformation rules which capture contextual factors in predicting correct word-class assignments (hereafter tags).

[9] http://www.cs.jhu.edu/~brill

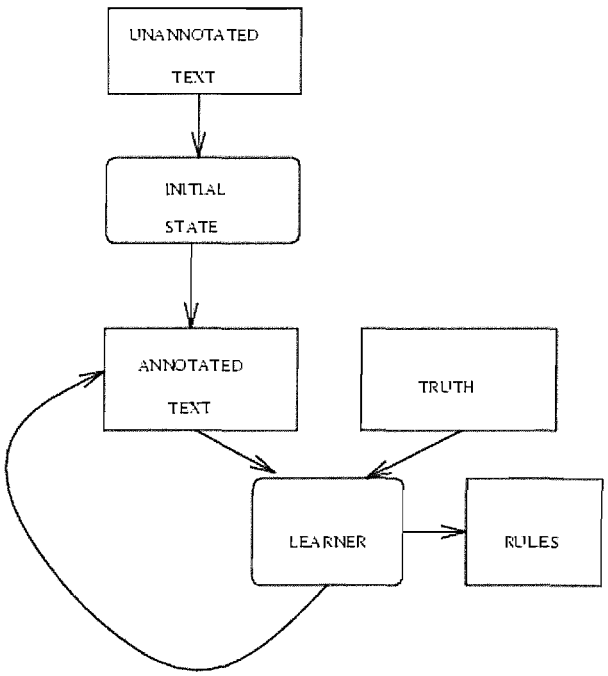

Figure 3-6 – Transformation-based error-driven learning [Brill, 1994]

An unannotated text is passed through an initial state annotator, the initial state annotator is applied only to unknown words (I.e. words not found in the lexicon) it assigns the default most likely tag, for English this is a proper noun tag for all words that start with a capital letter and a noun tag to all others. The known words are automatically tagged with the most likely tag from the lexicon. The output is a temporary annotated corpus which is compared to a goal corpus which has been manually tagged (the truth). The temporary corpus is passed through the learner. Each time the corpus is passed through, the learner produces one rule, the single rule that gives the best net improvement (since rules can introduce errors as well as correcting them) in tagging accuracy. Then the temporary corpus has the learnt rule applied to it and the process repeats as the temporary corpus is passed through the learner again. This process terminates when the net improvement falls below a pre-specified threshold (usually when the net change affects fewer than a few words). This produces an ordered set of transformation rules.

In fact the learning module produces two disjoint sets of rules as the learner actually consists of two stages, the lexical and contextual components. The lexical learner module

is used for deriving rules for tagging unknown words, whereas the contextual learner module is used for deriving rules for improving the accuracy (I.e. context rules for disambiguation). Both types of rule change one type of POS to another.

The space of possible transformation rules is fixed by a set of rule templates, which are essentially underspecified transformation rules. The unspecified values are instantiated to specific tags. Rules can also require specific words to appear in the template context.

The major problem of the Brill approach is its learning algorithm. As the transformations are data-driven, the vast majority of allowable transformations are not examined and certain rule templates are rarely used. The learner can generate ambiguous rules and word rules are usually less influential than tag rules. Also information cannot be learnt if there are no rule templates to capture it with.

The learning is a statistical approach rather than linguistically-driven. It cannot learn domain-specific rules, just generalisations on what it sees. The learning should not be based on frequency but by the quality of the disambiguation that it can obtain. What is required are rules based on disambiguation. This means that stronger disambiguating constraints need to be added to the lexical and context templates.

3.4.4. Treebank II Bracket Parser

The Penn Treebank Project[10] was founded by the Computer and Information Science Department at the University of Pennsylvania [Bies et al., 1995]. The Treebank II bracketing is designed to allow the extraction of simple predicate-argument structure. The bracketing marks out constituents and represents them in a hierarchical representation, as shown below.

```
(S (NP-SBJ Casey)
   (VP will
       (VP throw
           (NP the ball))))
```

[10] http://www.cis.upenn.edu/~treebank/home.html

Treebank II bracketing is a bottom-up approach to constituent parsing. There is only a small set of tags to represent the syntax but the combined tagset provides a higher level of constituency mark-up. There are 5 clause-level and 21 phrase-level tags that are integrated with the Brill POS tagset.

What makes the bracket parsing grammar unique is its approach to accepting all parses of a sentence. To be able to accept all sentences, without any rejection requires a more powerful mark-up than that used in conventional constituency grammars, bracket parsing therefore uses three tools, 'function tags', 'null elements' and 'pseudo-attachment'. It is the added use of these tools that extend the syntax and provide more effective annotations. The tools provide additional rules and annotator tags for the grammar.

Function tags are used to provide extra mark-up of individual constituents, phrases or clauses. The function tag aids in identifying and explicitly marking tacit and implicit knowledge from a sentence. These can be used in conjunction with each other to represent a rich mark-up of syntax. There are 20 function tags.

Null elements aid in providing structurally coherent parses. Null elements mark out where missing, but interpreted words or phrases should appear. This allows for easier identification of clause and sentence types, as the structures become more apparent. Null elements can also have co-indexing between the index and a referent, where one (or both) of the parts appear in an unorthodox position. There are 6 null element tags.

Pseudo-attachment is a method of showing that non-adjacent constituents are related. This defines the relationships in the overall parse tree, and makes sure that related parts are joined at the relevant levels. There are 4 pseudo-attachment tags.

The extended grammar caters for simple well defined constituent structures as well as for rich, intuitive, interpretations of more complex structures. This is beneficial in that it provides a powerful, almost human understanding of grammar and language.

The problem with the Treebank II bracketing is that it is all manually annotated and the domain coverage can be poor because the bracketing is on a specific corpus. In industrial use the expert knowledge of treebanking must be encapsulated automatically and be readily applied to a range of applications for it to be useful in Information Extraction.

3.4.5. Link Grammar Parser

The link grammar parser[11] created by Davy Temperley, Daniel Sleator and John Lafferty is a syntactic parser of English, based on link grammar, an original theory of English syntax. The grammar is manually encoded and gives lexically specific information.

Given a sentence, the system assigns to it a syntactic structure (see Figure 3-7), which consists of a set of labelled links connecting pairs of words. A link is formed if both words have complementary connectors. For example the link between the words 'the' and 'cat' is permissible because the word 'the' has a D+ link (where the + represents a D link must be made to the right of the current word), and the word 'cat' possesses the complementing D– connector (where the – represents a D link must be made to the left of the current word).

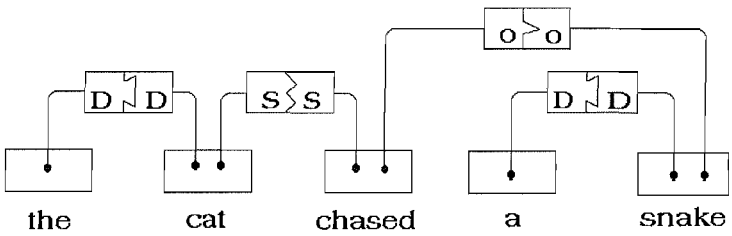

Figure 3-7 – Link grammar connections

A valid link adheres to both 'word' and 'global' rules. Every word has (at least) one rule in the grammar, which dictates the possible connections it can make in both directions. The ordering of the connectors is very important as it indicates the relative linear closeness of the words being connected. Globally the crossing of links is not permitted (the projectivity constraint of dependency language models) and all words must be indirectly connected. Both of these global rules stipulate a valid sentence.

There are constraints placed on the connectors to define extra specificity for a word. A constraint is denoted using a subscript, for example a D+ connector can form a link with a D– link but a Dx+ connector cannot link with a Dy–. The main constraints applied are for word agreement, for example a normal D link can connect a determiner to a noun, but a

[11] http://www.link.cs.cmu.edu/link

Ds link can link a determiner to a singular noun, and a Dp link can link a determiner to a plural noun.

Word sense ambiguity is resolved using connector subscripts. A word that has multiple meanings takes subscripts to disambiguate the alternate meanings. E.g. the word 'her' has two rules for disambiguation because it can occur as a personal pronoun 'her.p' as in 'I know her' and as a possessive pronoun 'her.d' as in 'her umbrella'. It is important that different meanings, however small, have different rules, as it aids in resolving ambiguities. This use of connector subscripts to resolve word sense ambiguity is simplistic and may not be accurate, for further reading of word sense disambiguation systems see Word Expert Parser [Small, 1980] and WordNet [Miller et al., 1990].

A further benefit of connector subscripts is that it alleviates the rigidness of the tagset size problem. This is important because most grammars focus too heavily on getting the correct tagset in order to map the language with, but in this case the tagset size is flexible as a new tag can be created by adding an extra subscript on a connector.

The power of the link grammar is having every possible word listed and to show the connections to other words. The burden of a complete lexicon is that things such as abbreviations, every inflection of every possible word, compound words, numbers, punctuation etc., all need to have rules, thus increasing the size of the ruleset and its complexity. If this is performed correctly and completely then the link grammar can easily be more powerful than any other syntactic parser, but the realisation is that this is not achievable. The link grammar therefore uses tools such as morphology to guess unknown words.

The link grammar is still only a syntactic parser, so it still has problems with semantics. It does take a step further than constituency parsers as it directly attaches together words in a particular relationship, but that is where it ends. For example, idiomatic verbs are identified in the grammar but they are clustered together by their tense and numerical agreement rather than their semantics.

After a syntactic structure has been established, the structure requires post processing to validate the generated links. Each sentence is divided up into domains based on its link

types. Then post processing applies domain constraint rules, which declares the validity of a structure based on the combinations of links present in a given domain.

Even after post processing, multiple parses can be expected if there are ambiguous words, unknown words or first-word capitalisation ambiguity in the sentence. In each event an exhaustive parse is required and the model with the least cost is chosen. The aim of the cost system is to select a parse from multiple ambiguous parses by the lowest cost. It is an aid in choosing a preferred connection.

Sometimes the link grammar cannot find a suitable parse, in order to allow for robust parsing, a null link system is incorporated. If the parser cannot parse a sentence normally, it tries ignoring one word in the sentence. The parser then finds all the linkages it can, ignoring that one word. If it still fails then the number of null links increases incrementally and tested again. In null link parsing the connectivity requirement is suspended. Therefore disconnected islands form.

3.5. Summary of Chapter

This chapter has provided a background history of Natural Language Processing (NLP), in particular a detailed discussion into the two polarised language models of constituency and dependency on which all Information Extraction systems are based.

Often the dependency model has been neglected by researchers but it was provided in the literature review as it helped to evaluate the strengths and weaknesses of both models.

This was followed by an overview of the parsers available for both language models. There was also a comparison of benchmark systems used in NLP for both types of model. The architecture of these systems helped to articulate a design for the author's theme extraction system.

4. Language Model Issues

This chapter provides an analysis of the constituency and dependency language models and the main issues that both systems generate.

The constituency model issues focus on the components that are used in constituency pipeline. The issues for each component are discussed, as well the generic weakness of all components which is the use of sub-standard components.

The dependency model issues highlight the weakness of using constituency components to build a dependency system and also an issue with dependency structure parsing.

Both models use a grammar to form syntactic structures and so the issue of the choice of grammar rules is also discussed. A summary concludes the chapter.

4.1. Constituency Model Issues

4.1.1. Analysis of Components in a Typical Pipeline

The constituency pipeline is a complete architecture consisting of tokenisation, morphological analysis, sentence splitting, POS tagging, syntax, semantics, discourse and pragmatics. This section describes each component and its issues.

4.1.1.1. Tokenisation

Tokenisation is the normalisation of the text by breaking up the input string into delimitations called tokens. The process of converting the input string to a token stream is performed by the tokeniser. The decision on where to split the input string and the assigning of attributes to the tokens is instigated by the tokeniser's ruleset.

The tokeniser splits the text into very simple tokens such as words, numbers and punctuation. The aim is simple recognisation of often repeating items and to enable greater flexibility by placing the burden of the work on the latter components which are more adaptable.

The default assumption is that an orthographic word (separated by spaces from the adjacent word) is an appropriate token. Although there are exceptions to this: a single orthographic word can contain more than one grammatical word. E.g. in the case of verb contractions and negative contractions such as 'she's', 'they'll', 'don't', 'isn't' etc. In these cases an attribute needs to be assigned to each grammatical word. Also quite frequent is the opposite circumstance where two or more orthographic words are given a single attribute. E.g. multiword prepositions such as 'up to' function as a single word. Naturally, whether such orthographic sequences are treated as a single word depends on the context.

The tokenisation problem branches out into trying to capture formal descriptions of everything, which is not what the author intends to do, but in cases such as finding multiword tokens, detection needs to be performed as early as possible to stem the flow of errors in latter components. So it is not only paramount that single tokens are captured, but also the net effect of combining tokens together to form larger entity types.

Another non-trivial problem is determining the tokenisation of times, dates, numbers and punctuation. E.g. 'twenty-second of Jan. '96' is semantically equivalent to '22/01/1996'.

Finally, the author does not expect the tokeniser to deliver any semantic information, because of the limited information context it works with.

4.1.1.2. Morphological Analysis

Morphological analysis is the process of recognising the root form of a morphological variant. Morphological variants can be of two distinct classes. Inflectional morphological variants share the same basic meaning and word-class as the root form, which is important in determining the word-class of unknown words. E.g. 'kick, kicks, kicked, kicking'. Derivational morphological variants can have a different meaning and different word-class as the root form. E.g. 'friend, friendly, friendliness, friendship'.

A morphology rule often strips a suffix or prefix from a word, and sometimes adds back replacement characters, to produce a possible root form [Porter, 1980]. Rules must be applied recursively as multiple derivations are common. Sometimes stripping prefixes and suffixes produces unavoidable errors. E.g. 'pretend' → 'tend', 'army' → 'arm'.

A system that analyses both prefixes and suffixes can generate an ambiguity depending on the order in which the prefix and suffix are applied. This is sometimes referred to as prefix-suffix interaction. For example 'undone' if interpreted as 'un' + 'done' means something that has not been done, another possible interpretation is the past tense of the word 'undo' and means the reversal of something previously done.

Another form of morphological analysis is lexical compounding. This is when two or more words are concatenated together to form a new word. The meaning and word-class for the compound is usually determined as a function of the individual words. Although this inference is not always correct. Consider the compound noun 'steppingstone'. This is an instance of non-literal language, because the head noun 'stone' is not the generic concept for the compound, by contrast compound nouns like 'limestone' and 'gravestone' are linked to their generic concept [Fellbaum, 1998]. Lexical compounding analysis often draws the wrong conclusions. However, it is sufficiently productive to make it an important analysis.

4.1.1.3. Sentence Splitting

Sentence splitting is the task of delimiting a sentence using final sentence punctuation such as periods, exclamation and question marks, white-spacing and quotations. To rule out errors caused by the use of abbreviations, a list of potential ambiguous candidates is dismissed. For example 'Dr. Smith' where the abbreviation 'Dr.' is considered a pseudo sentence split, similarly acronyms 'A.B.C.' are not contemplated as sentence ends.

A sentence splitter cannot rely purely on abbreviation lists to find incorrect sentence ends as all possibilities are never accounted for; it needs to be more intelligent about how it guesses correct sentences. For example a legitimate sentence starts with a capitalised word and ends with final sentence punctuation, whereas acronyms tend to be all capitalised and separated by periods.

Well formed sentences are required before applying the next stage of Information Extraction, that of part-of-speech tagging.

4.1.1.4. POS Tagging

POS tagging is a classification problem. The tagger takes a token as an input and must guess its tag class. Some tokens are unambiguous so are assigned a unique tag. Ambiguous tokens are assigned multiple tags. With multiple tags the relative probability of choosing a particular tag and the ordering of these tags is required to help choose the correct tag assignment for a token in latter components. This method is not guaranteed to be successful because of language idiosyncrasies.

In Natural Language Processing one of the major problems with POS tagging is choosing the correct tagset. In an ideal world this is the minimum set of tags that can incorporate every token in the language without ambiguity. There is research on tagset design [Elworthy, 1995] that proves that the relationship between tagset size and tagging accuracy is a weak one and is not consistent even when applied to the same language. The conclusion was to choose the tagset according to the requirements of a given application rather than to optimise it for the tagger.

The external criterion of the tagset is that it must be capable of making linguistic distinctions required in the output. Such that the number of classes in the tagset must be related to the distinctions needed by the syntactic structure imposed on the text. The internal criterion is making the tagger as effective as possible. Sometimes a single tag is used to cover multiple meanings such as transitive and intransitive verbs, since taggers cannot reliably predict the correct tag based on the context of only one, two or three words. These tags can then be handled trivially.

A small tagset should improve tagging accuracy as it puts less burden on the tagger to make fine distinctions. The number of decisions required is smaller and hence the tagger need contribute less information to make the decisions. The problem of this is that a later process using more context is required to disambiguate each word. On the flip-side of this argument is the use of a larger tagset to give extra specificity to a word, the cost of using this approach is the complexity of designing the tagset and the correct assignment of tags.

The improvements needed for better tagging techniques to determine ambiguous and unknown words are:

- Use linguistic knowledge of the language.
- Use a larger window scope to improve context information.
- Limit the class of tags that can be assigned.
- Morphology or any other way of identifying the surface form (I.e. inflectional analysis of unknown words).

4.1.1.5. Syntactic Analysis

Syntax is the parsing phase of the constituency pipeline, it builds up sentence structure and aids in the understanding of the parts of the sentence. Syntax disambiguates a whole sentence based on context information of other words and structures within the sentence.

The most expensive method is to parse all of the input with a natural language parser and grammar. The parser has complex grammar rules as it is trying to describe the syntax of the language. This requires being able to deal with larger scopes, bracketing of constituents, ambiguity resolution etc. The output is a hierarchical tree.

A less costly approach is to accumulate sequences of words that occur between words specified in a stop word list (a list of words that contain little information). The remaining sequences are typically simple noun or verb phrases. Occasionally there are words at the end of the sequence that must be removed to ascertain a correct phrase. A potential limitation of this approach is that the sequences are interpreted in isolation, without any understanding of the sentence context in which they occur. This means that local ambiguities cannot be resolved because of the lack of context.

Syntax though cannot resolve everything, for example it has problems with attachment ambiguities, mainly prepositional and conjunction. Prepositional phrase attachment is a common source of ambiguity in NLP, Some major work has been researched by [Brill & Resnik, 1994] [Collins & Brooks, 1995] [Nakatani, 1991] [Pantel & Lin, 2000] [Ratnaparkhi, 1998] and [Ratnaparkhi et al., 1994]. The goal is to attach the prepositional phrase as a classification of either an adverbial attachment (attaching to a verb) or as an adjectival attachment (attaching to a noun). The scope is often a 4-tuple.

The reason why it is a problem is that even in human classification of prepositional phrase attachments, the accuracy given the full context of a sentence is 93.2% and drops to

88.2% when only given 4-tuples (the results are based on three treebanking experts on a set of 300 randomly selected test events from the Wall Street Journal corpus).

It has been argued by Nakatani [Nakatani, 1991] that structural and lexical rules are not enough to disambiguate all kinds of prepositional phrase attachment, even though there is a close performance between humans and machines, and Nakatani uses pragmatics to resolve ambiguities.

4.1.1.6. Semantic Analysis

Semantics deal with interpretation and meaning, for example an idiomatic verb has semantic representation independent of the semantics of its constituents. I.e. the idiomatic verb 'scrape the bottom of the barrel' has the interpretation 'take whatever is left after the best has been taken'; and 'kicked the bucket' means 'died'.

At present formal semantics are not adequate for the richness of natural languages and also cannot deal with word sense ambiguities, so they do not deal with pragmatics. Resolving word sense ambiguities [Pantel & Lin, 2002] is important in language processing as it provides more disambiguation than that of the word form. A polysemous word can have multiple senses, for example, the word 'suit' has two senses, such as:

'Suit' → 'Jacket', 'Shirt' etc.
'Suit' → 'Lawsuit', 'Allegation', 'Case' etc.

Therefore not only does a word need to be identified, but also the correct sense needs to be recognised. Inventions of manually compiled dictionaries (E.g. WordNet), usually serve as a source for word senses, however they often include many rare senses while missing domain-specific senses.

One method of inferring the semantics of a word is to use its context. E.g. 'Tezguno makes you drunk' makes you think that 'Tezguno' is a beverage. The intuition is that words that occur in the same contexts tend to be similar. This is known as the Distributional Hypothesis. This hypothesis is not limited to words in similar contexts and can be applied to dependency relationships in dependency graphs [Lin & Pantel, 2001],

especially if two dependency relationships tend to link the same set of words, the hypothesis is that their meanings are similar.

Even when using semantics the results of a parse can still be ambiguous. Using the same problem of attachment ambiguities introduced in the previous section of syntax, one sentence can have multiple semantic interpretations. For example 'I saw the man with the telescope' where the first interpretation is 'the man with the telescope was seen by me', and the other is 'using a telescope I saw a man'.

Another non-trivial problem is idiomatic verbs (hereafter idioms). They are usually defined as a non-literal, fixed expression. An idiom is non-literal because its semantic representation is independent of the semantics of its constituents. An idiom is fixed in that the words that make up the expression appear next to each other in the correct order, so 'kicked the bucket' is not equivalent to 'the bucket kicked' or 'the kicked bucket' etc. However, the definition, as with natural language, is not strongly constrained.

The author argues that idioms cannot be easily recognised because non-literal and literal expressions are hard to disambiguate, and that they are not fixed expressions.

Idioms are a sub-category of phrasal verbs. These are a type of verb that function more like a phrase than a word. A phrasal verb can have two senses, as a literal meaning or as an idiom. For example consider the phrasal verb 'drop off':

'Drop off'	→	'Fall asleep' (Idiom)
		'Stop and give something to someone' (Idiom)
		'Decline gradually' (Literal)

It is apparent that even a simple expression such as 'drop off' has many ambiguous meanings, and even problematic, multiple idiomatic meanings. Once again this ambiguity resolution requires disambiguation using context. In the past context has been used to resolve single word syntax problems (e.g. POS-tagging) but in this case context needs to understand whole expressions and the actual semantics that they impose.

Idioms are not fixed because of two major issues. Firstly structural modification can exist within the expression see [Baldwin & Villavicencio, 2002] and [Ifill, 2002], and secondly an

idiom can contain morphology. For example reconsider the idiom 'kicked the bucket' in these particular contexts:

Modification	→	'Kicked the bloody bucket'
		'Kicked the holy bucket'
Morphology	→	'kicking the bucket'
		'kick the bucket'
Both	→	'kicking the bloody bucket'

All these examples mean exactly the same thing, they are all derivatives of the verb 'die' in different tenses, but are expressed uniquely. The only thing that remains fixed is the word-order of the idiom constituents, but because modification and morphology can affect the idiom, then it is difficult for a comprehensive lexicon of idioms to be stored.

4.1.1.7. Discourse and Pragmatic Analysis

According to Chan [Chan et al., 2000], discourse is understood to refer to any form of language-based communication involving multiple sentences or utterances. The most important form of discourse of interest to NLP is written text. Whilst text normally appears to be a linear sequence of clauses and sentences, it has "long been recognised by linguists that these clauses and sentences tend to cluster together into units, called discourse segments that are related pragmatically to form a hierarchical structure".

Discourse analysis goes beyond the levels of syntactic and semantic analysis, which typically treats each sentence as an isolated, independent unit. The function of discourse analysis is to divide a text into discourse segments, and to recognise and reconstruct the discourse structure of the text as intended by its author. It is assumed that discourse provides improved disambiguation where sentential analysis is insufficient because it is higher-level and encapsulates more context of the text.

The goal of discourse analysis, therefore, is to structure the linguistic context so that it can be understood as a cohesive sequence of sentences, in essence it is primarily interested in the correct ordering of sentences. Moreover, it needs to deal with inter-sentential relationships. Discourse is independent to language variation as it attempts to understand the linguistic context irrespective of how the information is written.

Discourse analysis can be viewed as the use of words and phrases to connect together separate clauses that are directly related to another to form a coherent story. It is important to determine which words and phrases are used as discourse clues and understand how to use them.

Ahrenberg [Ahrenberg et al., 1990] highlights a problem with discourse by stipulating that while there is, in the Artificial Intelligence literature, fairly large agreement on the usefulness of the notion of discourse, there are no simple algorithms for detecting discourse structure.

Carbonell [Carbonell, 1983] puts it well in that natural language discourse exhibits several intriguing phenomena that defy definitive linguistic analysis and general computational solutions. However, some progress has been made in developing tractable computational solutions to simplified version of phenomena such as 'ellipsis' and 'anaphora resolution'.

More recent research has aimed to integrate literature from discourse, pragmatics and psycholinguistics in order to design systems that better understand the intention of written text. A prime example of this is work by Aretoulaki [Aretoulaki, 1996, 1997] and [Aretoulaki et al., 1998], who reviews the major theories of language analysis and converts relevant hypothesis into an architecture that aims to capture surface, intermediary and pragmatic functions. Aretoulaki's work aims to integrate these functions by using the lower (surface and intermediary) levels to objectify the higher, more abstract (pragmatic) levels.

Often when the issue of discourse is broached then pragmatics is also mentioned. Pragmatics is the study of the aspects of linguistic structure affected by extra-linguistic elements – like the users of language and the time and place of utterance – that are relevant for language production and understanding.

In conclusion discourse analysis is useful in determining the relevance of a theme in a document as discourse uses the whole document as context. After some consideration, it was decided that pragmatic analysis is not suitable as it is more to do with intonation and speech rather than written text, which is not the main goal for this research.

4.1.2. Component Issues

Most researchers only focus on one area of the pipeline. This means that they use pre-developed components, by other researchers, in order to get a complete output from the pipeline. As a result sub-standard earlier components are used such as poor tokenisers. By using this type of approach some of the critical information required for theme extraction is already lost. Ideally the whole pipeline needs to be built from scratch, so that the boundaries and the work required of each component are known, and the flow of information from one component to the next are well understood. With all components their goals, input and output need to be clearly defined. At the end of the pipeline everything that needs to be captured is captured.

There is also a problem with error propagation. All components have an impact on the quality of theme extraction; ideally each component builds higher-level constituents whilst reducing ambiguities from earlier levels and passing on error free constituents to the next component. This is because error propagates from the bottom-up so every component depends on the accuracy of the lower components. Ambiguity in the pipeline is resolved by using larger contexts, which often requires latter rules to grow in complexity.

The final issue is that of commitment, where there is a lockdown of unambiguous constituents as early in the pipeline as possible. Then these unambiguous constituents can be used to aid ambiguity resolution.

4.2. Dependency Model Issues

The author's theme extraction process will primarily focus on the constituency model. However to complete an overview of the language models, the issues for dependency models are discussed below.

4.2.1. Building on Constituency Components

Early computational linguists worked with dependency grammars but only in forms that were easily convertible to constituency grammars, and can essentially be parsed with the same techniques. This meant that constituency components were used to build dependency grammars. For example POS taggers are commonly used in dependency grammars because they can revert to tag information when information for a particular word is not available.

As observed in [Nivre & Nilsson, 2003] many systems developed for dependency parsing use algorithms that are straight forward modifications of the algorithm used for constituency parsing. Faced with massive ambiguity and non-determinism, these algorithms rely on dynamic programming and tabulation to derive compact representation of multiple analyses with reasonable efficiency, but there is no attempt to resolve ambiguities or remove non-determinism. Similarly entire papers [Xia & Palmer, 2001] are devoted to converting between constituency and dependency models, even though in the opinion of the author, the models are polarised.

Frequently the dependency grammar is biased to make the parser more simplistic. Most dependency parsers compromise the model by not using a full lexicon, as creating the knowledge is a laborious task, and the given language constraints are not powerful enough to map natural language.

It order to bring out the full potential of dependency grammar as a framework for natural language parsing, the author needs to explore alternative parsing algorithms.

4.2.2. Dependency Structure Parsing

Dependency parsing also has its problems. Presently the majority of the dependency world is limited to projective dependency grammars, although it seems clear that certain constructions in natural language are non-projective in nature.

Dependency parsers also fall into well-known problems having to do with attachment ambiguity for prepositional phrases and other adjuncts [Nivre, 2003]. This is because most of the parsers choose the closest possible link to attach to, which of course, may not be the correct decision. There are also problems with valence violations, linking across syntactic boundaries and errors caused by elliptical constructions:

- *Valence Violations* – It is difficult to impose restrictions on the number of dependents of a single head.

- *Linking Across Syntactic Barriers* – Despite the fact that most dependencies are local to the syntactic clause[12], there is nothing that prevents the parser from adding links across clause boundaries. Valence and linking errors can be corrected by making the parser sensitive to the number of dependents of a given head, and to the presence of clause boundaries in the input string.

- *Errors Caused by Elliptical Constructions* – Dependency grammar presupposes that all syntactic constructions have a head. However in elliptical construction it is often the case that the expected syntactic head is omitted. Cases in point are clauses without a finite verb and noun phrases without a head noun. These errors require major changes either in the grammar or in the parsing algorithm or both.

[12] The word 'clause' used in this sense, means all the dependents of a particular headword, clauses do not exist in dependency grammars because the linking is word-to-word

4.3. Grammar Issues

Traditionally, grammatical models have been constructed by linguists without any considerations for computational applications, and later by computationally-oriented scientists who have first taken a parseable mathematical model and then forced the linguistic definition into the model which has usually been too weak to describe what a linguist desires.

There requires a balance of linguistic-driven data and mathematical formalism. Dependency grammar is both descriptively adequate and formally explicit [Järvinen & Tapanainen, 1998]. Whereas a constituency grammar is too limited given its reliance on mathematical formalisms such as context-free grammars. Extra linguistic properties need to be incorporated to make constituency grammars more powerful.

One major aspect of a grammar model is the ordering of its rules. Poor rule ordering can have many disastrous effects such as thrashing because of poor rule interaction, erroneous overlapping of rules because of the need for robustness and the production of redundant rules which never fire.

Finally their needs to be a decision made on the variety of rule types to be used in the grammatical model. It is hypothesised that with more types, the better a grammar is at capturing expressive information.

4.4. Summary of Chapter

This chapter has provided an analysis of the issues of constituency and dependency language models. Often the dependency model has been neglected by researchers, but the author found it necessary to compare the issues surrounding both models.

The typical components used in a constituency pipeline were analysed and their issues were raised. The main weakness in the constituency model was the use of sub-standard components. This is because researchers often implement benchmark systems into their own system which lose information and propagate errors. This can speed up implementation time but the author argued that the systems be built from scratch.

The dependency model highlighted the weakness of using constituency components to construct a dependency system rather than building a full grammar for the system. Using constituency components is the easy alternative, but the main power of a dependency system is its grammar, which is difficult to implement, but the use of constituency components only served to detrimentally weaken the dependency model.

The main issue found with both language models was that of ambiguity. In the constituency model the ambiguity flowed through the whole length of the pipeline affecting each component. In the dependency model the main ambiguity stemmed from the use of constituency components to build dependency structures.

Both language models use a grammar to form syntactic structures, so the choice of grammar was also discussed.

5. Design

The constituency language model has been used to design the author's theme extraction system because the model allows for modularisation of natural language components, which shows more transparency in their interactions. However the grammar which is the main strength of a dependency language model is also utilised as a component in the constituency model with as much syntactic and syntactic function labelling as possible to increase the performance of theme extraction.

This chapter provides the justifications for the choices of the proposed system. The problems of traditional architectures are highlighted and the issues are used to justify the requirement for a new, lossless architecture, which provides benefits such as minimising information loss and error propagation.

This lossless architecture requires a new formalism for representing information which is referred to as a quad. The components that are used in the architecture are described, which contains the purpose of a component and the tasks it performs. This is followed by the issue of computational feasibility of a lossless architecture and its implementation. A summary concludes the chapter.

5.1. System Architecture Overview

5.1.1. Traditional Architecture Problems

Traditional language models use deterministic methods to make decisions along the pipeline. The problem with this is that decision-making components usually lead to component filtering which propagates a loss of information. Also many systems re-use benchmark systems that are not the most accurate, such sub-standard components introduce errors. The combination of losing information and error propagation creates a deterioration of the quality of information that is propagated by the architecture.

An architecture has been researched by the author to the extent of fully understanding the components required for theme extraction. Fundamental to this architecture is an ethos for lossless information.

5.1.2. Requirement for a Lossless Architecture

5.1.2.1. Minimising Information Loss

The backbone to the architecture and of its components is based on the idea of lossless information. This is the retention of a complete information history for all the components in the architecture and is utilised as a fully referenceable lookup of information produced, and the component and rule that created that information.

The aim of the architecture is to keep all the decisions, and pass them on through other components, so that each component has all the information at hand and can make a more accurate evaluation of all the alternatives.

A lossless architecture means that all possibilities are tried rather than just a subset. It is in affect an exhaustive, brute-force approach. A benefit is that no decisions have to be made which means that pruning is not performed. This component filtering is when a component performs pruning on the input which means that information is lost before it arrives at the next component. When this occurs the value of the information in a pipeline slowly degrades and is lost. A lossless architecture stops the pitfalls of component filtering.

Take for example a traditional part-of-speech tagger which assigns a word-class to a word. If the word is ambiguous then the tagger makes a decision to assign the most likely word-class to it, and ignores the rest of the choices but the chosen word-class can be incorrect. The proposed lossless architecture keeps all the choices and pass them on, so that latter components can make a more educated guess given more information.

Apart from preventing component filtering a lossless architecture can also transparently show the interaction of the rules and components. These interactions become more visible when all the information history is available. The rule history leads to improved component interaction. Understanding the component interaction minimises the contention of rules between components, by making rules independent from one another to reduce rule-overlap, so that information flows easily from one component to another. Understanding the component interaction also contributes to maximising the utilisation of

a component as it shows the information history of other components so allows the information produced from earlier components to be used in latter components.

A lossless system needs to deal with exhaustive data. The author has not chosen to make efficiency an issue, so it is not a factor to be considered. However in practice this is too computationally expensive. There are time considerations to take into account when using a brute-force approach and this rarely fits into a company's strategy.

5.1.2.2. Minimising Error Propagation

Information persistence is required so that information is always preserved and so that error propagation can be detected and reduced by latter components.

By keeping all the information, and also having provenance information about each rule, the architecture preserves a history of rule selection. This improves the design of the rule interactions as the affect of each rule becomes transparent, and any underperforming rules that are identified can be modified for effectiveness. This enforces transparency in the architecture and minimises error propagation by reducing the error passed from one component to another.

5.2. Quad Formalism

The lossless architecture keeps track of all choices using quads. A quad is a novel way of representing a slice of information and a tool for enforcing the underlying concept of lossless information. A quad is formalised as follows:

(Attribute , Value , Confidence Value , Provenance Rule)

A quad has a traditional attribute-value pair but it is novel because it also has a confidence value associated with the pair and also provenance information which shows which rule and component created the quad. These are discussed below.

5.2.1. Attribute-Value Pair

An attribute is a feature that needs to be preserved. Each component produces a different set of attributes, and each rule in a component produces a specific attribute. For example the tokeniser component can produce attributes such as 'word', 'number', 'punctuation' etc.

A value is the instantiation of the attribute-value pair. It is usually a specific instance of the attribute. For example attribute is 'number' and value '60'.

5.2.2. Confidence Value

The confidence value is given to the quad based on the likelihood that the attribute-value pair is assumed to be correct. It is the perceived trust in the pair being correct. The confidence value ranges from 0-to-100%. A zero-value means that there is no trust, whilst 100% assumes the opposite.

A zero-value is required so that incorrect quads can be determined. This helps in searching for errors in components and rules. Another reason for giving quads a zero-value rather than to remove them is that it retains information persistence.

At 100% confidence value the quad is be assumed to be committed. I.e. further components assume that the quad is always correct.

The strategy for deciding the confidence value is statistical in nature, but if it is subjective then it is usually proportional to the rule context size and / or intuitiveness of the rule.

The confidence value is of major importance when attempting to resolve ambiguity. Attribute-value pairs can be given different confidence values to order a set of ambiguous quads.

5.2.3. Provenance Rule

The provenance marks the origin of the rule that created the quad. The provenance shows the component that created the rule and also the rule itself. This information is valuable in maintaining a full information history and can suggest changes in components and rules that create inappropriate quads.

5.2.4. Quad Set

Typically every token in a document has information stored about it as a set of quads (henceforth quad set). The quad set is initially empty. Each component in the pipeline creates new quads for the token. A component analyses the token's information in its quad set and then produce new quads based on which rules fire in the component. The new quads are then appended to the quad set.

Storing of all quads means that even incorrect quads can still add value to the system as it allows the system to detect errors and allows recovery to a point where a quad was assumed correct.

5.3. Components in Architecture

The architecture chosen is modular. This is so that each component has distinct input and output boundaries and has unique task descriptions. The modularisation in essence provides a balance of work for each component. It also allows for more transparent component interaction.

The components in the architecture are knowledge engineered, developed using experience of language and making use of human intuition. The components can be rule and statistical based. Development of the components is time consuming and changes are hard to accommodate.

To preserve the lossless ethos the components are built from scratch as often benchmark systems perform pruning on the output. To this end each component produces and preserves information. The architecture is shown below (see Figure 5-1).

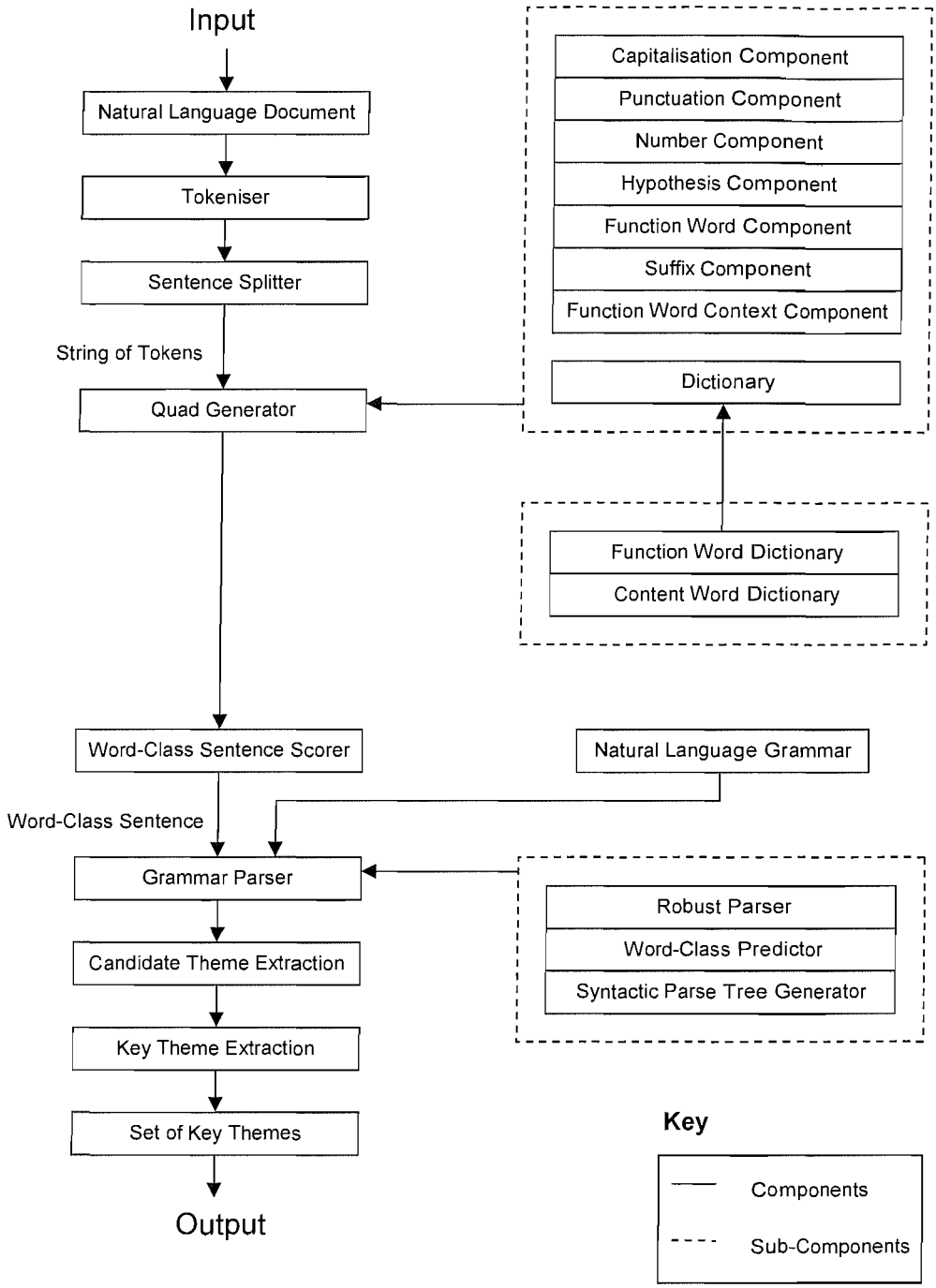

Figure 5-1 – System architecture

The further down the component pipeline the larger the contexts that the components work with. The early components work on single or multiple tokens whilst the latter take

sentence and document contexts. This is because latter components handle more complex processes and also use the larger contexts to provide better disambiguation.

The following sections justify the choice of components and highlight their task description.

5.3.1. The Input

A discussion of natural language texts (see 1.1.3. Natural Language Texts) concluded that a written document can be dependent on an individuals writing style, use of grammar and extent of their vocabulary. It also mentioned that a document can be unstructured and have ambiguities throughout. The challenge of natural language texts ensures problems when attempting to interpret the meaning of the document and extracting only key themes from it. A theme extraction system must be able to resolve these ambiguities.

The input to the system is a collection of natural language texts, so that the system can be tested in the real-world. These texts were made purposely harder to interpret by making them only text-based. This collection is referred to as a corpora of plaintext documents.

Plaintext has been deliberately chosen because of their difficulty and challenge to interpret and so theme extraction is a harder problem. Plaintext documents contain only text and no metadata tags so a more detailed understanding of linguistics is required to understand the documents.

If the system is successful then it can be applied to any type of document as the technique does not rely on metadata and just functions on the text.

5.3.2. Tokeniser

The tokeniser is the first component in the architecture for theme extraction. It is required for delimitation of lexical units so that they can be used in latter components.

It is often overlooked by language engineers as a simple peripheral component. Nevertheless the tokenisation process has a significant impact on the rest of the components if it is incorrectly performed as errors propagate from this first component.

The task of the tokeniser is to recognise the largest possible units that have internal integrity and have some likelihood of subsequent occurrence [Woods, 1997]. Considerable effort is required to define a default behaviour that is correct in the widest possible variety of foreseeable circumstances.

5.3.2.1. Delimitation

The tokeniser initially delimits a plaintext document by whitespace to form lexical units often called tokens. The whitespace is often formatting such as space, tab, and newline characters. The delimitation process destroys the original structure of the document, and produces instead a stream of tokens.

5.3.2.2. Character Splitting

The next stage of tokenisation sees the stream of tokens undergo character splitting. This is the splitting of all non-number and non-word characters from both sides of a token so that multiple compound parts of a token are split. Splitting off the characters from both sides reduces the complexity of a token, especially for problem tokens such as those that contain quote marks or brackets, it also provides more information for latter components. For example a token is split into multiple tokens as follows:

('$5.6m') → (' $ 5.6m ')

Tokens that consist of purely punctuation and symbol characters are exempt from character splitting as they do not have any number or word characters. If this is the case then the token is ignored.

Tokens with internal punctuation such as hyperlinks (e.g. http://www.news.bbc.co.uk) are ignored because punctuation is split from the extremities of the token, so the whole hyperlink is preserved.

Problems occur if an apostrophe occurs at the end of a token or if the token has multiple punctuation. For example if an apostrophe is split off a token (e.g. Joss' album) then it becomes ambiguously either a single quote or a possessive marker. If it was originally a possessive and becomes a quote then single quote tracking becomes distorted.

Tokens that have multiple punctuation and in particular punctuation that functions as a sentence end, can complicate matters. For example consider tokens such as 'continuing...' or 'stop!?!', if the character splitter strips all punctuation then each punctuation character becomes a sentence end which is incorrect.

5.3.2.3. Contraction Expansion

A token can be a contraction. A contraction that uses an internal apostrophe needs to be recognised as a grammatical compound of two words. The original token is replaced with the expansion. Using the expansion removes the apostrophe and makes tracking of single quotes and possessive markers easier. It also makes the original token easier to interpret as it becomes two grammatical words rather than a complex token.

If a token is a candidate for contraction expansion, regular expressions are used to pattern match the token to its grammatical interpretation. Contractions can be of two types, verb and negative.

A verb contraction is always expandable to two words. The first word is a personal pronoun. It is the text string of the token up to the apostrophe. The apostrophe and the remaining text string is the contraction of the second word, the second word is either an auxiliary or modal auxiliary verb.

The contraction can be ambiguous, however the choice of expansion does not have any significant affect, as the ambiguous options are all the same word-class either auxiliary or modal auxiliary verbs.

Special disambiguation rules are used for the contraction "s' as it can be either a word or a possessive marker. The list of verb contractions and their expansions used in the system are shown in Table 5-1.

Verb Contraction	Expansions
'm	am
're	are | were
've	have
'd	could | should | would
'll	shall | will
's	does | has | is | was

Table 5-1 – Verb contractions and expansions

A negative contraction also expands into two grammatical words where the second word is always 'not'. Typically the 'n't' text string of the original token is the expansion, and the rest of the original token forms the first grammatical word. There are two exceptions to this which are the words 'can't' and 'won't'. These special cases possess their own rules and are tested first as they are more specific instances of negative contraction.

Negative Contraction	Expansions
n't	not
can't	can not
won't	will not

Table 5-2 – Negative contractions and expansions

5.3.3. Sentence Splitter

A sentence splitter is required to take the stream of tokens produced by the tokeniser and to build up sentences. This sentence context is accepted as being large enough to provide useful information and small enough to be used computationally. All the following components in the pipeline use this sentence context.

There are generally three acceptable sentence delimiters which happen to be punctuation marks they are periods, exclamation and question marks. In this system an exclamation or question mark automatically ends a sentence. A period though can be ambiguously used as either a sentence end, or part of an acronym or abbreviation. Context rules are used to determine the correct choice.

An acronym is typically a proper noun phrase that has been shortened into a single token where only the initial character of each word in the phrase is kept and stored separated by periods. The characters in the acronym are usually either lower or uppercase letters.

An abbreviation is typically a contracted form of a proper noun. Therefore they usually are a capitalised or lowercase word followed by a single period. An abbreviation lexicon is used by the system as a resource to resolve ambiguities. This is a lexicon of commonly used abbreviations that can produce false sentence endings. There are 96 abbreviations in total, it is not assumed to be complete lexicon but contains common cases.

The period punctuation can be interpreted as an ambiguous choice between sentence end and sentence continuation. This happens when the period is part of an acronym or an abbreviation and the following word is capitalised. The strategy chosen is to end the sentence as long as the following token is a capitalised function word. If not then continue the sentence. For example:

'J.C. Penney'	→	sentence continuation
'First Boston Corp. A sole underwriter'	→	sentence end

If a period is not part of an acronym or abbreviation then it must be a sentence end.

5.3.4. Quad Generator

The quad generator's task is to assign information quads to tokens in a sentence. These quads are used in latter components. The quad generator performs lexical analysis of each token as such most of the rules are linguistically based. It uses a series of components to perform these analyses. The components are:

- Capitalisation component
- Punctuation component
- Number component
- Hypothesis component
- Function word component
- Suffix component

- Function word context component

5.3.4.1. Capitalisation Component

The capitalisation component provides information on possible tokens that can be classed as nouns.

The capitalisation component checks the likelihood of a token being a capitalised word and assigns it a quad if it believes it is correct. It also assigns a quad for certain types of acronyms and abbreviations.

5.3.4.2. Punctuation Component

The punctuation component catches non-word and non-number information that can be useful in latter components.

The punctuation component checks the likelihood of a token being a punctuation and assigns it a quad if it believes it is correct. It also assigns a quad for the possessive marker and some symbols.

Punctuation that serve an important syntactic purpose are given their own class, all others are given a generic punctuation class. For example the symbol '&' can be interpreted as a conjunction, e.g. 'H&M'.

5.3.4.3. Number Component

The number component catches number information which is useful in latter components.

The number component checks the likelihood of a token being a number and assigns it a quad if it believes it is correct. There are many types of numbers so this component has an assortment of rules to capture the information. E.g. apart from numerical digits, punctuation such as commas (order of thousand, million etc.), periods (decimals), colons (time or ratio), forward slashes (dates) and hyphens (number-word modification) can be used in determining numbers.

The system does not use a number lexicon so textual numeric representations (e.g. sixteen) are not be recognised by the system, and also some complex numeric representations (e.g. 1.3e6) are not recognised by the current rules.

5.3.4.4. Hypothesis Component

Essentially the hypothesis component speculates the word-class of a token based on its attributes held in previously created quads. This provides extra word-class information for a token when linguistic cues are seen in the text.

This component takes the quads produced by components that do not produce word-class attributes, and hypotheses a word-class based on the information it sees. Currently there are hypothesis rules for attributes from the capitalisation and number components.

5.3.4.5. Function Word Component

This component performs a lexical lookup of a token to see if it is a function word, and if it is then a set of word-class quads are given to the token. The aim of the component is therefore to provide each function word with an associated set of word-classes. Only function words are used because they are a closed vocabulary. The lookup is case sensitive as this sensitivity preserves more information.

The set of word-class quads is created using two resources a lexicon of function words and a frequency lexicon created from the Wall Street Journal (WSJ) corpus that contains the function words and the relative frequencies of that function word occurring as a specific word-class. A function word can ambiguously have many different word-classes. The confidence value of each word-class quad is linked to the relative frequency of the function word occurring as that word-class. The quad's word-classes and confidence values can help aid in statistical disambiguation in latter components.

As the frequency lexicon has been created from the WSJ the actual quads are domain dependent. There is also a mapping process that takes WSJ tags and converts them to word-classes that can be used in the system. Also for a word-class of a function word to occur as a quad it must occur at least 0.1% of the time. For example the function word 'A'

(lowercase 'a' produces a different set of quads due to case sensitivity) has 3 different WSJ tags of DT, NNP and NN, and each has different frequency of occurrence:

'A'	DT	→	1142
'A'	NNP	→	59
'A'	NN	→	10

The quads created are:

(Determiner	A	94.3	Initial_POS_Rule)
(Proper_Noun	A	4.9	Initial_POS_Rule)
(Noun	A	0.8	Initial_POS_Rule)

5.3.4.6. Suffix Component

The suffix component checks the likelihood of a token being a particular word-class based on its suffix. This is based on the hypothesis that the end suffix ultimately provides the word-class. It performs morphological analysis by analysing token endings to see if its suffix provides a useful discriminator of word-class information. Its aim is to speculate useful word-class information from unknown tokens.

The suffix component works over content words (aka open-class words). Words which match a suffix rule produces a word-class quad, these quads contain an open-class attribute, either of adjective, adverb, noun or verb.

A suffix rule consists of the suffix, the discrimination rate and the word-class. E.g. 'al 78.7 Adjective'. There are a total of 73 rules.

The suffixes used are amalgamated from work by Quirk [Quirk et al., 1985], the Porter Stemmer [Porter, 1980], and linguistic information. This is a class of suffixes that are strong discriminators of word-class.

The suffixes range from 2 – 4 characters in length. This range was chosen because a single character does not provide good discrimination rates, and more than 4 characters over specifies the suffix and leads to a loss of robustness in matching tokens. When

matching a word, the rule with the most specific suffix is selected because longer suffixes tend to provide higher discrimination rates.

The minimum word-class discrimination rate for each suffix is 50%, the mean is 86.2% and 43 rules are above 90%. The discrimination rate becomes the confidence value for the quad.

The suffix rules were analysed over the WSJ corpus. So they are domain dependent and are not applicable to all corpora.

5.3.4.7. Function Word Context Component

This component performs word-class disambiguation of function words. It performs a lexical lookup of a token to see if it is a function word (or part of a multi-token function word) and runs a series of associated rules. Each rule functions independently so for every rule it matches a word-class quad is produced. The aim of the component is therefore to provide each function word with an associated set of word-class quads. Only function words are used because there is a limited amount of them so rules can be designed for every single function word.

The rules are classified by word-class, and further sub-classified as either generic or specific rules. Generic rules typically apply to all the words in that word-class or a group of words. Specific rules typically apply to a group of words or a single word.

Every function word has a unique set of rules associated with it. These rules are linguistically based using real-world knowledge and are manually created so that they better articulate natural language. Each rule has a central position which is the function word, and linguistic context which requires matching. The linguistic context can be absent in some rules, but in the others the context can exist on either or both sides of the central position.

The linguistic context acts as a constraint to further disambiguate the likelihood of a function word being correct. This context is matched to the context surrounding the function word in the token stream. The context is only limited by the sentence delimitations. If the linguistic context is matched then the rule creates a quad.

The linguistic context is made up of specific words, word-classes, punctuation marks, regular expressions (e.g. zero or one, zero or more), and logical operators (e.g. OR, NOT). The combination of which provides powerful contextual constraints to increase function word disambiguation.

The confidence value for the quads has been assigned manually. The objective factor for the value is dependent on the quantity of linguistic context for the rule. E.g. no context usually gives a low confidence value, whilst context on both sides of the central token gives higher values due to the specificity of the constraints. A difference of this component to the others is that the confidence value can be negative, which means that the word in question is probably not a function word.

5.3.5. Dictionary

A dictionary is an important part of a natural language system as it is used to associate a token with a word-class. It is used to compensate for rule scarcity of the quad generator. This word-class information is useful in latter components.

A dictionary looks up a token and provides it with a quad for every word-class it is associated with. E.g. the word 'beat' has quads for adjective and verb classes.

There are two dictionaries, a function and a content word dictionary. These dictionaries are lexicons of words and associated word-classes. The function word dictionary is assumed to be complete as new words are rarely introduced into a language, whilst the content word dictionary aims only to be minimal in nature but provide a high impact for word-class assignment.

The aim was not to build complete dictionaries, but just to form minimal ones that benefit the system by being able to allocate word-class information to tokens, so that more accurate syntactic parse of natural language documents is possible.

5.3.5.1. Function Word Dictionary

The function word dictionary stores both single and multi-token function words. There are 314 unique single-tokens and 141 unique multi-tokens. Most of the words are used

ambiguously. The multi-tokens are particularly important as they determine the function of a group of words. This helps to resolve ambiguity of words by committing a group of words, and produce a more accurate syntactic parse.

The function word dictionary is linguistically produced. It is used primarily by the quad generator component. The words in the dictionary are particularly useful as they are the most frequently used words in a language and supply grammatical information on how sentences are structured. Also they are domain-independent which means that they can be used in any corpora without any bias. An obscure benefit is that function words cannot be speculated using morphological analysis so it is essential that they are clearly classified in a dictionary.

5.3.5.2. Content Word Dictionary

The content word dictionary only stores single-tokens. The aim was for a minimal lexicon that provides a high impact for word-class assignment. A complete lexicon cannot be achieved because content words are often introduced into a language. It is important to have some minimal content word dictionary as a function word dictionary by itself is not sufficient for producing accurate syntactic parses, and function words provide only grammatical information and little meaning, so content words are necessary to supply semantic information.

The content word dictionary was produced by analysis of the Wall Street Journal (WSJ) corpus. The resulting lexicon of adjectives, adverbs and verbs became the dictionary. The size of each is adjectives 3,353; adverbs 284; and verbs 3,221. Most of the words are used ambiguously. The dictionary size can be reduced if the lexical entries were stemmed. This was not carried out though as it can have created errors during conflation. In comparison dictionaries used in other systems are far larger in size, for example Brill's part-of-speech tagger uses a lexicon, also trained from the WSJ, which contains 70,697 lexical entries.

There are a few major limitations, firstly the dictionary was learned from the WSJ so it is domain-dependent, secondly the dictionary is assumed to be incomplete, and thirdly there is no lexicon for nouns.

A lack of a noun lexicon means that the system must compensate by being adaptable to speculate on tokens that function as nouns. Conversely the content word dictionary is greatly reduced as it does not have to cater for this ever growing word-class.

5.3.6. Tag-Set

The quad generator and dictionaries produce word-class information. These word-classes are used in latter components to produce syntactic parses from which to extract themes from. In particular, the grammar rules which build the syntactic structures are completely dependent on the word-classes.

The collection of word-classes is known as a tag-set. Design of a tag-set is a non-trivial problem as it is the basis of interaction between a majority of components. Ideally a tag-set provides unique tags for all classes of words that have a distinct grammatical behaviour. This way a tag can be used for disambiguating information.

A small tag-set often increases accuracy of word-class assignment as it reduces the chance of tagging inconsistencies because of fewer tags, but it produces tags that are too generic to capture linguistic detail and often these tags become ambiguously used. Conversely a large tag-set leads to complexity issues, redundancy and sparse data problems. Tag-set size can vary greatly, some example tag-sets and sizes are:

Brown	→	87, which has proliferated into:
Wall Street Journal	→	48
Lancaster – Oslo / Bergen	→	135
Lancaster UCREL	→	165
London – Lund	→	197

The choice of tag-set size always bears a cost. For example the WSJ is based on the Brown tag-set with syntactic and lexical conflations. The cost of this is that some of its tags are ambiguous, with distinctions having to be made by human annotators. For example the tags IN, NN, and VBG are used ambiguously as:

IN	→	Preposition or subordinating conjunction
NN	→	Noun, singular or mass

VBG → Verb, gerund or present participle

In the author's system the tag-set size is 67. These tags are for content words, single-token function words, multi-token function words, possessive marker, punctuation markers, and numbers.

5.3.7. Word-Class Sentence Scorer

At this stage in the pipeline every token in a sentence contains zero or more word-class quads. These quads are used for further processing at the syntactic level. If a token has zero word-class quads then a class of 'unknown' is assigned to it.

The word-class sentence scorer generates and priorities word-class sentences from the previously created quads so that they can be used to build syntactic structures.

The generation process produces every permutation of word-class sentences, based on each token in the sentence and the set of word-classes associated with it.

To prioritise each generated word-class sentence requires three phases. The first phase evaluates the score of each word-class. This requires the confidence value and provenance rule of the quad for that word-class. Each provenance rule has a weight associated with it. The score of the word-class is the product of the confidence value and the rule weight.

The rule weight depends on the component provenance. Each component in the architecture can be classed as either intuitive, statistic or linguistic. For example the hypothesis component is intuitive as it speculates the word-class of a token.

The rule weighting of the components has intuitive as the lowest and linguistic as the highest. This is because statistical analysis is preferred over intuitive assumptions, whilst linguistic is preferred over all as it uses the most context.

So in summary, each word-class score is the product of its confidence value and the rule weight which is associated with its provenance rule. If it is 'unknown' then the score is zero.

The second phase of word-class sentence prioritisation scores the entire word-class sentence as the sum of all the individual word-class scores.

The final phase prioritises the word-class sentences by a best-first approach. The highest scoring word-class sentence is the first to be syntactically parsed.

5.3.8. Natural Language Grammar

A grammar is possibly the most important part of a natural language system as it formalises how a language is used. The aim of a grammar is to provide a formal description of natural language using syntactic and linguistic information. This requires an understanding of the structures that are used in natural language. The grammar highlights the quintessential word-class sequences that occur with each other to form typical sentences.

A grammar is a set of rules that apply structure and word order to a language. The grammar rules used in the author's system has been derived from multiple sources including language books by Quirk [Quirk et al., 1985], and Biber [Biber et al., 1999]. There are 222 grammar rules.

An ambiguous grammar is allowed so that the expressiveness of natural language can be mapped. This means that grammar rules are often ambiguous. This ambiguity keeps with the lossless information ethos as all possibilities are explored rather than a simple deterministic choice.

The grammar provides a large coverage of natural language. It caters for both frequently and non-frequently occurring structures. It is the most descriptive and expressive component in the system. It is the first component that uses sentence, clause, phrase and word-level structures, all of which are related in hierarchical fashion. The highest syntactic level is the sentence and the lowest is the word-class. Higher-level structures are formed by lower-level ones. Each level covers many specific and generic structure types. As well as syntactic information the grammar also provides syntactic functions to supply extra richness such as subject and object.

A grammar rule is formalised as having a non-terminal label on the left-hand-side (LHS) followed by a colon followed by a right-hand-side (RHS) that can be empty, or consist of a mixture of non-terminals and terminals. A non-terminal used on the RHS of a grammar rule expands to the grammar rule(s) that has the non-terminal as its LHS label. The terminals used in the grammar are the word-classes for content and function words, punctuation markers and numbers. A terminal is enclosed by single-quote marks whilst non-terminals are not. The top grammar rule is known as the 'S' rule.

The grammar allows free word order by allowing rules to be cyclic in nature (i.e. when non-terminals are repeated in other grammar rules), and having optional structures (i.e. when a non-terminal can occur zero times).

Grammar rules decide the boundaries of structures. A rule determines how a structure is formed and also its word-order. A rule can be visualised as a series of specific linguistic constraints, the more expressive the rule the more constraints it contains.

5.3.8.1. Dictionary and Natural Language Grammar Size

A dictionary and a grammar are perhaps the most important components in a natural language system for theme extraction. In most systems they tend to be produced by statistical analysis of manually annotated corpora. The author has approached the problem from a linguistic perspective and both the function word dictionary and natural language grammar have been knowledge engineered. However it was noted that the resulting system did not provide enough word-class information for accurate syntactic parses so the content word dictionary was designed to resolve this issue.

A major problem with both dictionaries and grammars are their size. The author has performed work to minimise the size of these components whilst maximising their impact. The formation of the content word dictionary and improving the coverage of the grammar were both obtained by use of a flexi-dictionary.

A flexi-dictionary is a component that was used to learn and resolve weaknesses in the architecture. Its main contribution to the system was to produce a minimal content word dictionary (see 5.3.5.2. Content Word Dictionary) and improve the coverage of the grammar from analysis of the Wall Street Journal (WSJ).

In the original system only a function word dictionary was used along with linguistic and statistical components to determine word-class information. The word-class sentences produced contained many 'unknown' tokens, which were mainly due to content words not receiving word-class assignment because of a lack of rules. It was valuable at this stage to produce more word-class information for content words prior to parsing. A flexi-dictionary was used to increase the number of word-classes in a word-class sentence.

The flexi-dictionary applies a word-class of either adjective, adverb, or verb over the WSJ to see what affect it has on the output.

The system analyses the WSJ for each word-class. The word-class sentences produced by the system are analysed by the flexi-dictionary. For each 'unknown' token in the word-class sentence the corresponding tag in the WSJ corpus is found, if the tag is the same as the word-class being analysed then the 'unknown' is substituted with the word-class (see Figure 5-2). The actual 'unknown' word and its following word-class context window are stored.

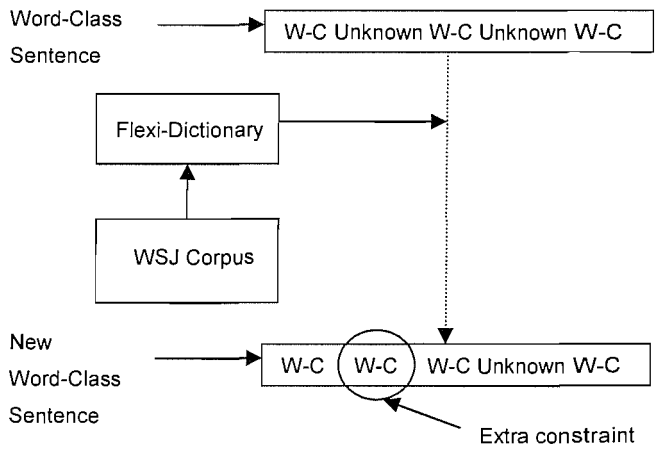

Figure 5-2 – The flexi-dictionary component

After the entire WSJ is analysed the set of stored words becomes the content word dictionary for that word-class, and the set of word-class context windows is used to improve the coverage of the grammar.

5.3.9. Grammar Parser

The grammar parser component takes as input a word-class sentence and parses the sentence using the natural language grammar and a non-deterministic parser. The output is a set of syntactic parses from which to extract themes from.

Deep grammar parsing using non-deterministic components such as the grammar and parser is beneficial to theme extraction as it shows all possible ways of breaking a sentence into grammatical constituents and show the relationships between them which is useful for resolving structural ambiguity.

It is an improvement over deterministic parsers that apply a best-fit analysis and lose vital information, and over partial parsers that perform shallow Information Extraction so does not produce relationship information between the extracted constituents.

The non-deterministic parser used in the theme extraction system is described in the next section.

5.3.9.1. Earley Parser

The Earley parser was choicen as the non-deterministic parser. It is the only component in the system that was not built from scratch. Nevertheless it is well suited with my lossless architecture ethos as it retains all information it produces.

Like the other parsers discussed in section (3.3.1.4. Non-Deterministic Parsers), it can use either an unambiguous or ambiguous grammar and simulate parallel execution of multiple copies of a left-right parse, and so simulate non-determinism.

The Earley parser performs a systematic search to explore the space of possible syntactic parses of a word-class sentence based on the natural language grammar. A word-class sentence can either parse or fail. If it parses it builds a data structure called a parse graph to store all the syntactic parses.

The parser starts by traversing the top grammar rule, the 'S' rule, and expands through any non-terminals until it reaches rules that expect a terminal, a pointer is then placed

before the terminal of that grammar rule. This set of rules expecting a terminal can be viewed as rules 'in-flight'. Next the parser tries to consume the first word-class from the word-class sentence, if this word-class matches the expected terminal in a rule in-flight, then that rule is continued and the pointer for the rule moves past the terminal, and the word-class is taken as consumed. A rule in-flight is discontinued if it cannot consume the word-class. If all the rules in-flight are discontinued then the parse for the word-class sentence fails. The consumption of a word-class therefore acts as a constraint over the set of possible rules in-flight. The process repeats by using the pointers in the rules in-flight to generate a new set of rules in-flight for each subsequent word-class in the word-class sentence. The sentence parses successfully if the entire sentence is consumed and the top grammar rule has been matched.

Another view of the process is that each rule in-flight is a possible syntactic parse, and only the parses that consume the next word-class are allowed to continued, and all others are discontinued. In this view the word-class constrains the set of possible parses.

Formally, an Earley parser is successfully if it achieves the goal of moving from the start state 'S: * ...' to a final state 'S: ... *' after consuming the entire sentence. Where 'S' is the top grammar rule and '*' represents the pointer position in a rule. The parser uses the natural language grammar rules and the following pseudo-code for the functions predictor, scanner, and completer:

```
If (the pointer is not at the end of a grammar rule)
{
  If (the next element is a non-terminal)
  {
    Run Predictor
  } Else
  {
    Run Scanner
  }
} Else
{
  Run Completer
}
```

The parser creates a new set of rules in-flight for each new word-class of the input sentence. Remember that a terminal is enclosed by single-quote marks whilst non-terminals are not.

- *Predictor* – If the pointer is not at the end of a rule (i.e. 'X: ... * ...') and if the next element is a non-terminal use the predictor. This produces a new rule in-flight. The new rule always has a pointer at the start of its right-hand-side. E.g. The grammar rule 'X' expands the non-terminal Y to form a new rule in-flight 'Y'.

  X: * Y
  Y: * 'a'

- *Scanner* – If the pointer is not at the end of a rule (i.e. 'X: ... * ...') and if the next element is a terminal use the scanner. If the next word-class in the sentence matches the terminal then the word-class is consumed and the pointer for the rule is moved past the terminal. E.g. The grammar rule 'X' consumes the terminal 'a'.

  X: * 'a' Y
  X: 'a' * Y

- *Completer* – If the pointer is at the end of a rule (i.e. 'X: ... *') then use the completer. Which completes the rule and its parent rule now has the non-terminal completed. E.g. The grammar rule 'Y' completes after consuming the terminal 'a', so its parent rule 'X' can now complete its 'Y' non-terminal.

  X: * Y Z
  Y: * 'a'
  Then terminal 'a' is consumed
  Y: 'a' *
  X: Y * Z

To demonstrate the parser, a trivial example is used that has an ambiguous grammar. The input it uses produces two syntactic parses. I.e. S → X [a] and Y [b c], and S → X [a b] and Y [c].

Input: a b c

Grammar: S: X Y
X: 'a'
X: 'a' 'b'
Y: 'c'
Y: 'b' 'c'

In Table 5-3 below the 'Current Sentence Position' is shown by a '*'. In the 'Rules In-Flight' column, the numbers in brackets define provenance information. The first number is the rule identifier and the second number is the parent identifier. This is necessary to keep track of ambiguous parses.

Current Sentence Position	Rules In-Flight	Comments
* a b c	(0) S: * X Y (0) (0) X: * 'a' (0) (0) X: * 'a' 'b' (0)	The initial set of rules in-flight is produced. The top grammar rule 'S' is expanded until all rules in-flight expect a terminal. This requires the predictor. As the grammar is ambiguous there are two 'X' rules.
a * b c	(1) X: 'a' * (0) (1) X: 'a' * 'b' (0) (1) S: X * Y (0) (1) Y: * 'c' (1) (1) Y: * 'b' 'c' (1)	The scanner consumes the terminal 'a'. Rules not expecting the 'a' are discontinued. The rule 'X: 'a'' completes. The parent rule 'S' moves past the 'X' non-terminal. New predictions are made for the non-terminal 'Y'.
a b * c	(3) X: 'a' 'b' * (0) (3) Y: 'b' * 'c' (1) (3) S: X * Y (0) (3) Y: * 'c' (3) (3) Y: * 'b' 'c' (3)	The scanner consumes the terminal 'b'. Rules not expecting the 'b' are discontinued. The rule 'X: 'a' 'b'' completes. The parent rule 'S' moves past the 'X' non-terminal. New predictions are made for the non-terminal 'Y'.
a b c *	(5) Y: 'b' 'c' * (1) (5) Y: 'c' * (3) (5) S: X Y * (0)	The scanner consumes the terminal 'c'. Rules not expecting the 'c' are discontinued. Both 'Y' rules complete. The 'S' rule completes. The parse is successful and the parse graph contains 2 ambiguous parses.

Table 5-3 – Earley parsing example

5.3.9.2. Limitations of the Earley Parser

The author was forced to make a few major improvements to the Earley parser so that it met necessary design requirements in order to produce improved syntactic parse trees for theme extraction. The improvements are a robust parser, word-class predictor and parse tree generator.

The Earley parser fails to parse a sentence if there are insufficient grammar rules to provide the necessary coverage for a sentence. The likelihood of failure is high unless the sentence is short and simple. The parser has been modified so that it can continue to parse a sentence even after fail points. This way a parse is given to the entire sentence even if some parts of the parse are fragmented and unconnected. This robust parsing means that the parser now accepts any input sentence, rather than just valid sentences as it did before.

Often an input sentence can have 'unknown' classes, which means that no components in the pipeline had been able to speculate a word-class of a token with any authority. To allow parsing of these unknowns requires modification of the Earley parser to allow it to predict a set of word-classes for the token based on the set of rules in-flight.

The original author of the Earley parser has stated that the parse graph is "near impossible to traverse". However the author has designed a parse tree generator for traversing the parse graph to produce syntactic parse trees. The parse tree generator is capable of producing syntactic parse trees for any input sentence.

The robust parser, word-class predictor and parse tree generator are discussed in the next few sections.

5.3.9.3. Robust Parser

One of the consequences of an incomplete grammar is that a parser sometimes fail with a given input. If a full parse of a sentence is not possible then a robust parser is used. A robust parser is required to make sure that as much syntactic information is extracted from the input as possible. A robust parser is an acceptance that a grammar is

incomplete. It is assumed that a perfect grammar can parse all natural language sentences making robust parsers obsolete as there are no fail points.

In a set of rules in-flight, if there is no rule that can consume the next word-class of input, then a full parse fails. This occurs because none of the rules can form a constituent given the current state of the parses and the next word-class. The parse graph before the fail is analysed and a strategy is used to extract the best syntactic structures. After the fail point, the input used to build the syntactic structures is considered consumed, and a parse restarts using the top grammar rule with the remaining input.

The robust parser strategy selects syntactic structures using a series of constraints, which in order of precedence are:

- *Rule Completer* – This means that the whole of a grammar rule must be matched in order to be completed, this is to prevent partial matching of rules which weakens the linguistic constraints that a grammar imposes. A completed rule also means a linguistic structure is produced with the correct syntactic boundaries.

- *Highest Non-Terminal* – This is a choice of parse amongst a set of parses for the input. The structure with the highest non-terminal is chosen. The order of precedence is an 'S' parse, followed by parses with other non-terminals and the lowest is a single-token. This is based on the assumption that it is better to have fewer syntactic parse fragments for an input, so the ideal is a single full parse and the nightmare scenario is all single tokens.

    - *'S' Parse* – An 'S' parse is the ideal. An 'S' parse whether it is for a full parse or a robust parse functions the same. All ambiguous options for syntactic parse trees are kept.

    - *Other Non-Terminal Parse* – Sometimes the input successfully parses but it does not form a 'S' parse, nevertheless the information stored in the parse graph still provides enough information to produce syntactic parses. A strategy is used to choose the syntactic parse trees from the highest non-terminal. Some redundant information is removed.

- *Single-Token Parse* – This is the final resort. If the input fails for all lengths then all that remains is a single-token. This occurs because the token cannot combine with its surrounding context to form any structures. This is because there is no suitable grammar rule.

- *Longest Match* – This is the longest consumption of input that produces a valid syntactic parse. If the parse is invalid, then one token from the end of the input is removed until the input parses successfully. The preference for longest match is because it contains the most context. The longest match criteria is formally known as a back-tracking left-to-right longest match.

The constraint of rule completion is never relaxed as formation of consistent bracketing is fundamental for creating accurate syntactic structures. The other two constraints can be relaxed. The order of relaxation begins with the longest match meaning that a shorter input can be attempted for a parse, followed by compromising on the highest non-terminal which means searching down the order of precedence for a parse.

5.3.9.4. Word-Class Predictor

If a token is not associated with any word-classes, then it is given a class of 'unknown'. These 'unknown' tokens require special processing during parsing.

When an 'unknown' is parsed it assumes the set of expected terminals generated by the rules in-flight. Each terminal is a candidate word-class that the 'unknown' token can be given based on the state of the parses and the grammar rules. The task of the word-class predictor is therefore to generate a set of word-classes that an 'unknown' token can be given the input and the grammar.

In essence, an 'unknown' functions as a unconstrained terminal as all the rules in-flight are accepted and all options are explored simultaneously so it does not constrain the set of possible syntactic parses. Therefore the more unknowns that exist in the input the weaker the constraints on the proliferation of syntactic parses.

The issue that arises is that predicting unknowns becomes a grammar over-generation problem. To make some arrears the set of generated word-classes is limited to the open

and closed word-classes and does not allow punctuation markers, symbols, numbers or possessives to be candidates for the 'unknown' token.

5.3.9.5. Syntactic Parse Tree Generator

The author of the Earley Parser, Luke Palmer, warns "don't try to use the parse "graph" yet, as it is near impossible to traverse". However, the only way of producing the syntactic parse trees was to attempt this challenge. The author has successfully managed to traverse the parse graph and produce syntactic parse trees. The parse tree generator is the component that performs that task.

It can produce the syntactic trees for either a full or robust parse, and for unambiguous or ambiguous parses.

5.3.9.6. Limitations of using the Grammar and Parser

There are fundamental problems of using the grammar and parser. Although a grammar can be useful in formalising a language, it can suffer from incomplete language coverage and contain rules that do not necessary represent idiosyncratic usage of language. Also the grammar specifies only syntactic information and presents no semantic information that might be useful in semantic analysis. The parser applies the grammar to only a single sentence so this limited domain may restrict the information that may be gleamed from using a larger document context.

The reason for using the grammar and parser is to preserve the flow of information. Both components are non-deterministic so all possibilities can be explored by the theme extraction component. Another option would be to use a purpose built linguistic parser such as GATE. The GATE Information Extraction system, ANNIE, produces figures of between 80-90% precision and recall on news texts [Cunningham et al., 2002a]. It should be noted that GATE uses large noun lexicons in which to identify entities of interest, and there are no such lexicons in the author's system. However, the main issue for not using a system such as GATE is that it uses deterministic components that propagate information loss. Another reason is that GATE uses a shallow parser which does not preserve deep structure so relationships between grammatical constituents cannot be used to resolve structural ambiguity.

5.3.10. Theme Extraction

The theme extraction component is the final component in the pipeline. It uses the syntactic parse trees produced for each sentence to extract key themes for the entire document. Theme extraction is split into two processes, the first extracts the candidate themes from each sentence, and the second is to extract key themes for a document based on the set of candidate themes.

5.3.10.1. Candidate Themes

Candidate themes for a sentence are extracted from the syntactic parse trees. Theme extraction is therefore dependent on the grammar and the parser. The process extracts structures with labels that can be potential theme indicators. These labels include noun phrases, and specific types of subjects, objects and prepositional complements. This provides an objective approach for theme extraction.

A sentence can form multiple syntactic parses. Each parse provides its own set of themes for a sentence. The aim of choosing a single set of candidate themes for each sentence therefore requires techniques for conflating all the ambiguous set of themes for that sentence. A single set of candidate themes simplifies the final process of choosing key themes. Below is a list of the steps taken in chronological order:

- *Removal of Non-Essential Modifiers* – To start with, all non-essential modifiers are removed from each candidate theme. Pre-determiners and determiners do not add value to a theme so these modifiers are removed from the front of all candidate themes. E.g. 'the red car' → 'red car'.

- *Ubiquitous Themes* – If a theme occurs in every parse tree and has the same syntactic label, then it is automatically accepted in the set of candidate themes.

  For all other themes there are three techniques to conflate them into the single set of candidate themes. First duplications are removed then analysis is performed for unique and intersected themes.

- *Duplicated Themes* – All exact duplications of a theme are merged into one. So if a theme exists with various syntactic labels because their parse trees were

different, then all the variations are merged into one syntactic label of 'ambiguous'. This ambiguous classification is required so that it clearly shows that the theme used to have multiple syntactic functions. An instance of the duplicated theme is accepted in the set of candidate themes.

- *Unique Themes* – A unique theme has no intersection with other themes, so it is a theme on its own merit. A unique theme is accepted in the set of candidate themes.

- *Intersected Themes* – With intersected themes the aim is to choose the best single representation. This is required so that only one theme represents the intersection of themes. The choice is the theme that completely consumes the other themes. I.e. 'average life of the certificates' consumes 'average life' and 'certificates'. So the latter two themes become redundant. This encompassing theme is kept because the other sub-themes are derivable from it so no information is lost.

  If there is no true consuming theme then intersected themes are combined into a longer theme that encompasses the themes. I.e. 'rich duke' and 'duke of York' becomes 'rich duke of York'.

  An encompassing theme is given the syntactic label of 'derivation' because the theme is derived from a collection of other themes. An encompassing theme is accepted in the set of candidate themes.

5.3.10.2. Key Themes

The key themes of a document are generated based on an analysis of the candidate themes for each sentence. Each candidate theme is independently scored using variables for precedence, length, and frequency. These factors aid in identifying key themes. Below is a list of the factors, followed by the equation used to score each theme, and how the themes are ranked:

- *Sub-Themes* – The author assumed that a theme containing function words can be split so that sub-themes can be extracted. This is based on the assumption

that function words act as syntactic glue between potential themes. These sub-themes are extracted and scored as candidate themes on their own merit. They are given the syntactic label of 'sub-theme'.

- *Theme Precedence* – The set of syntactic labels that are used for theme extraction is listed in Table 5-4. Each label is assigned a weight. The weights are a way of boosting the importance of different types of themes.

Syntactic Label	Weight	Comments
Ambiguous	1	An ambiguous label shows that the syntactic label cannot be determined for that theme. It means that the theme had multiple types of syntactic structure in the parse trees. Therefore it is weighted the lowest.
Derivation	1	A derivation label is used for an encompassing theme of a set of intersecting themes. As it is a derived theme then there is a chance that the original themes had multiple syntactic labels, therefore a derivation is weighted the same as an ambiguous label as the syntactic label can not be determined for the derived theme.
Noun Phrase	2	A noun phrase is the default theme. It does not serve any syntactic function.
Object	3	An object is a good indicator of a theme.
Prepositional Complement	2	A prepositional complement is weighted the same as a noun phrase as it does not serve any syntactic function.
Subject	4	A subject is a strong indicator of a theme.
Sub-Theme	1.5	A sub-theme label is used for shorter themes derived from longer themes. They are given a lower weight than most syntactic labels as their syntactic function is unknown.

Table 5-4 – Syntactic label weights

- *Theme Length* – The system has a preference for longer themes, which means a preference for theme phrases rather than single-word themes. The theme length is a sum of the number of tokens in the theme. The impact of the theme length is the least significant of the factors as both short and long themes can be key themes.

  To normalise theme length, any themes that contain function words are penalised. A sum of the number of function words in that theme is used as the penalty during scoring.

- *Theme Frequency* – This is the frequency of a theme occurring in the document. The theme must be matched in its entirety but the matching is case insensitive because the same theme can have different case depending on where it occurs

in the document. Theme frequency is considered to be a major factor in identifying key themes.

The problem of using theme frequency is that the match needs to be exact however as the system has no component for stemming then similar themes cannot be matched.

A candidate theme score is calculated as:

$$\text{Candidate Theme Score} = \frac{\left((\text{Synatactic Label Weight} \times \text{Theme Length})^{(\text{Theme Frequency}+1)}\right)}{(\text{Number of Function Words}+1)}$$

Equation 5-1 – Candidate theme score

Theme frequency and the number of function words both need to be non-zero so their values are incremented by one.

After the candidate themes are scored, they are ranked, highest-first. The key themes are those that have a score exceeding a threshold. A maximum of the top 20 key themes are shown for evaluation purposes. Of course, less are available if their scores do not meet the threshold.

5.4. Computational Feasibility

A lossless architecture is able to improve the accuracy of theme extraction whilst minimising the propagation of error. However, this preservation of information has a detrimental affect on computational feasibility as all permutations are tried rather than a best-first approach.

As mentioned before this research is aimed at quality rather than speed, but some memory problems have manifested into the system. The main problems arise from the sizes of the dictionary, tag-set and natural language grammar. The size of each has been kept to a minimum whilst aimed at providing a maximum impact for theme extraction. The design of each is non-trivial as they have interactions with the other components in the architecture.

This section quickly highlights where the author has had to make a decision to speed up certain components so that the system does not run out of memory. Unfortunately the cost of these shortcuts is that the ethos of the lossless architecture is not kept which means that there is a reduction in the potential power of these components and ultimately vital information is lost.

5.4.1. Word-class Sentence Scorer

The word-class sentence scorer generates and priorities word-class sentences from the previously created quads. The generation process produces every permutation of word-class sentences, based on each token in the sentence and the set of word-classes associated with it. This component sometimes runs out of memory.

A shortcut is used to produce the top word-class sentence rather than generate all possible options. A potential problem is that the word-classes in the sentence are incorrect.

5.4.2. Grammar Parser

The grammar parser forms syntactic parse trees given a word-class sentence. As the parser is non-deterministic it generates all possible permutations. This component sometimes runs out of memory.

If the number of syntactic parse trees produces exceeds a threshold of 50,000 then it becomes too computationally expensive to perform theme extraction from all this trees. A shortcut is used to produce the first syntactic parse tree rather than generate all possible trees. This dramatically reduces computation, as some sentences can produce incredible numbers of trees because either the word-class sentence has many 'unknown' tokens, so there is less context to constrain the proliferation of trees, or it has many punctuation tokens which act as weak constraints.

A potential problem is that the syntactic parse tree does not have the correct structure so theme extraction is inaccurate, this is a major problem. Also as there is only one tree then 'ambiguous' or 'derivation' themes that are used in theme scoring are not identified.

There has been one occasion where the memory problem occurred whilst actually producing the parse graph for a sentence. Unfortunately at this stage it is impossible to determine the number of syntactic parse trees that are likely to be produced. So the system cannot provide a shortcut for this potential problem. The problem was with sentence 4 (after sentence splitting) of file 'DjVu – ACO S Range'. This sentence has since been removed prior to evaluation.

5.5. Implementation

The code for the system was written entirely in ActiveState Perl[13] version 5.6.1. This programming language was used for its powerful text manipulation, rapid prototyping and access to the Comprehensive Perl Archive Network[14] (CPAN), which contains useful modules necessary for the system.

All the components in the system were written from scratch apart from the Earley parser, which was modified to provide the necessary requirements for use in the system. The Earley parser and the other modules incorporated from the CPAN are:

- *Parse::Earley 0.15 (Luke Palmer)* – An implementation of the Earley parser.
- *Text::Balanced 1.95 (Damian Conway)* – A module that is used by Parse::Earley to extract delimited text sequences from strings.
- *Test::Simple 0.54 (Michael G Schwern)* – A set of basic utilities for writing tests.

The hardware used to evaluate the system was an Intel Pentium 3 processor, 850 MHz, 384 MB RAM. The system runs very slowly due to the lossless architecture.

[13] http://www.activestate.com

[14] http://www.cpan.org

5.6. Summary of Chapter

This chapter has provided the justification of the choices for an automatic key theme extraction system. In particular the need for a new lossless architecture to overcome the problems of traditional architectures. The key benefits of a lossless architecture are the minimisation of information loss and error propagation.

The lossless architecture required a new formalism for representing information which was called a quad which contained an attribute-value pair, confidence value and provenance rule.

An architecture was implemented by the author that adhered to an ethos of lossless information. To preserve the lossless ethos the components were built from scratch as often benchmark systems perform pruning on the output. To this end each component produced and preserved information.

The design for the architecture used primarily Natural Language Processing (NLP) techniques to deliver an Information Extraction solution for natural language texts. This approach was chosen rather than a statistical approach as these systems cannot articulate human language idiosyncrasies. Even statistical systems that learn over a large corpora may not be able to build an accurate model of language. The NLP approach has been chosen as it was assumed that it can provide a better understanding of human language.

The main component in the architecture was the grammar. This provided the rules for syntactic structure and ultimately theme extraction, and has been the most difficult component to build. One of the goals for this research was to produce a grammar that can be used by Active Navigation. The value of the grammar to Active Navigation is affected by the cost of this grammar development.

The lossless architecture cannot be used in a commercial context as there is a consideration for speed, which leads to a trade-off between performance vs. speed. However computational feasibility was not an issue for this research as computational power constantly increases, so speed became less of a detrimental factor, and what was paramount was the performance of the lossless architecture.

6. Evaluation

6.1. Introduction to Chapter

This evaluation chapter provides an assessment of the performance of the author's theme extraction against that of Active Navigation's (AN) theme extraction.

The performances that are evaluated are the ability of the author's system to extract candidate themes from a plain-text document, and also the ability to extract key themes from a plain-text document.

The candidate theme extraction is measured against human annotation of themes. The key theme extraction is measured against both human annotation of themes and AN theme extraction.

The performances are measured by Information Extraction metrics of precision, recall and F-measure.

The aim of the research was to compare the set of key themes produced by the system and by AN, and then to evaluate the strengths and weaknesses of both systems. The evaluation of theme extraction is therefore only between the system and AN. The human annotation process is required to produce a gold standard from which to compare the two systems with.

The evaluation therefore omits from comparisons with other academic work, or with industrial equivalents.

6.2. Outline of Chapter

The chapter begins with a history of Information Extraction evaluation, including its origins from TIPSTER and the Message Understanding Conferences. This section provides the scoring strategy and metrics for which the evaluations are based.

The next three sections describe the approach used for human annotation of themes; the process of Active Navigation theme extraction; and the process of the author's theme extraction.

This is followed by the evaluations of candidate theme extraction and key theme extraction. The final two sections summarise the results and provide a summary of the chapter.

6.3. Information Extraction Evaluation

6.3.1. History of Information Extraction Evaluation

6.3.1.1. TIPSTER

Information Extraction systems have been evaluated under the support of Defense Advanced Research Projects Agency (DARPA) and other government agencies for almost a decade.

The TIPSTER Text Program[15] was a DARPA sponsored project that encouraged the advancement of state-of-the-art technologies for text handling through cooperation of researchers and developers in Government, industry and academia.

The aim was to provide the intelligence community with improved operational tools.

To improve text handling, TIPSTER focused on three metrics-based evaluations. Those of Information Extraction, document detection, and summarisation.

During the first phase of TIPSTER (1991 – 1994), the participants made major advances in creating the algorithms for Information Extraction and in improving the techniques for measuring the advances, through Message Understanding Conferences (MUC).

The project successfully concluded in 1998, an archive[16] is available.

6.3.1.2. Message Understanding Conferences (MUC)

Since early 1990, the MUC[17] evaluations have been funding the development of metrics and statistical algorithms to support government evaluations of emerging Information Extraction technologies.

---

[15] http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/overv.htm

[16] http://www.nist.gov/itl/div894/894.02/related_projects/tipster/

[17] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/index.html

In the mid-nineties MUC evaluations began to provide prepared data and task definitions in addition to providing fully automated scoring software to measure system and human performance.

The task definitions grew in complexity. At the beginning the tasks were to produce a database of events found in newswire articles from one source. More recently, the tasks were to produce multiple databases of increasingly complex information extracted from multiple sources of news in multiple languages.

The databases now included named entities, multilingual named entities, attributes of those entities, facts about relationships between entities, and events in which the entities participated.

The results of these evaluations were reported at conferences during the 1990s. The last of the conferences was MUC-7[18].

6.3.1.3. MUC-7

All the participants in the conference develop software systems which perform natural language understanding tasks. The systems are evaluated based on how their output compares with the output of human linguists. The MUC scoring software is for that comparison [Douthat, 1998].

The evaluation of the Named Entity in English task is focused upon, as this is most similar to theme extraction.

The evaluation begins with the distribution of the task definition and training set. The test set is distributed later, with results due after than. The domain for training was airline crashes and the domain for testing was launch events. Both the training and test sets contained 100 articles with human-generated answer keys from New York Times News Service corpora.

[18] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_proceedings/overview.html

The Named Entity task definition required the systems to mark-up the names of organisations, persons, locations, mentions of dates and times (relative and absolute), and direct mentions of currency and percentage.

Fully automated software is then used to evaluate the system. When the scorer is run, it reads in an output file with the human-generated answer keys, and an output file with the system's responses. The scorer aligns objects in the answer key file with objects in the system's response file. It then calculates various scores based on how well the responses agree with the keys. The scorer then tallies and reports metrics and error types for developers and evaluators in the form of score reports.

For Named Entity in English, the systems can achieve scores that are close to human performance [Chinchor, 1999]. The human annotators were still significantly better than the systems (see Table 6-1).

Systems	F-Measure Low	F-Measure High	F-Measure Average
Automated	69.67%	93.39%	84.13%
Human	96.95%	97.60%	97.28%

Table 6-1 – Automated system and human performances over the test set, where F-measure has precision and recall weighted equally

The next couple of sections explain the scoring strategy and metrics used to determine the performance scores.

6.3.2. Scoring Strategy

A scoring program is required to determine the scoring category for each system output (henceforth system themes) compared with the human-generated answer keys (henceforth gold themes). The following scoring categories originate from MUC [Chinchor, 1992]. They have been modified to deal with evaluation of candidate and key theme extraction.

- *Correct* – When the system theme and the gold theme occur over the same span of text in a document.

- *Partial* – When the system theme and the gold theme share a common span of text in a document. Sometimes a gold theme can encompass more than one system theme, or vice versa (see Table 6-2). Either way, all the involved themes still only score as one partial match.

Partial Types	Example. Format: 'gold theme(s) → system theme(s)' Multiple themes are separated by '+'
Gold completely subsumes system	surface drainage system → drainage
System completely subsumes gold	asphalt → drainage of the asphalt
Gold and system are intersected	flow performance → performance for any site
Gold completely subsumes multiple system	Specification of S Range channel units and accessories → Specification of S Range + units and accessories
System completely subsumes multiple gold	suffix DF + description → suffix DF to the description
Gold and system are intersected by multiple themes	efficient surface drainage → efficient + drainage of the asphalt

Table 6-2 – Partial types used in scoring

- *Missing* – This applies only to gold themes. A gold theme is missing if it does not occur in the set of system themes.
- *Spurious* – This applies only to system themes. A system theme is spurious if it does not occur in the set of gold themes.

These scoring categories are used by metrics to produce performance scores. The next section details each of the metrics.

6.3.3. Metrics

Metrics are pre-defined measures of performance calculable by comparison of system themes with the gold themes. In other words it is a measure of how well the retrieved information matches the intended information.

The metrics that are commonly used in Information Extraction have been influenced by the MUC competitions. As there is a strong connection between Information Extraction and MUC, often the MUC evaluation metrics of precision, recall and F-measure [Chinchor, 1992] tend to be used for Information Extraction. These metrics have

originated from the closely associated field of Information Retrieval and extended for MUC [van Rijsbergen, 1979].

Precision and recall measure different aspects of performance. The third metric, the F-measure, is a combined score for the overall performance.

Each of these can be calculated according to 3 different criteria – strict, lenient and average[19] [Cunningham et al., 2002b]. The reason for this is to deal with partially correct themes in different ways.

- *Strict* – Considers all partially correct themes as incorrect (i.e. they are categorised as either missing or spurious depending if the theme was either a gold or system theme respectively).
- *Lenient* – Considers all partially correct themes as correct.
- *Average* – Allocates a half weight to partially correct themes (i.e. it takes the average of strict and lenient).

The author used the 'average' criteria for partially correct themes. This is the most frequently used criteria in calculating performance scores. The metrics for precision, recall and F-measure are detailed in the following sections.

6.3.3.1. Precision

Precision is a measure of the ability of a system to present only relevant themes.

The precision metric[20] [Cunningham et al., 2002b] is a measure of the number of correctly identified themes as a percentage of the number of themes identified (see Equation 6-1).

In other words, it measures how many of the themes that the system identified were actually correct, regardless of whether it also failed to retrieve correct themes.

[19] http://www.gate.ac.uk/sale/tao/index.html

[20] http://www.gate.ac.uk/sale/tao/index.html

The higher the precision, the better the system is at ensuring that the themes that it extracts are correct.

$$\text{Precision} = \frac{\text{Correct} + \frac{1}{2}\text{Partial}}{\text{Correct} + \text{Spurious} + \frac{1}{2}\text{Partial}}$$

Equation 6-1 – Precision

6.3.3.2. Recall

Recall is a measure of the ability of a system to present all relevant themes.

The recall metric[21] [Cunningham et al., 2002b] is a measure of the number of correctly identified themes as a percentage of the total number of correct themes (see Equation 6-2).

In other words, it measures how many of the themes that should have been identified actually were identified, regardless of how many spurious themes were found.

The higher the recall, the better the system is at extracting all the correct themes.

$$\text{Recall} = \frac{\text{Correct} + \frac{1}{2}\text{Partial}}{\text{Correct} + \text{Missing} + \frac{1}{2}\text{Partial}}$$

Equation 6-2 – Recall

6.3.3.3. The Importance of Precision and Recall

The metrics of precision and recall are used to evaluate the automated systems.

The reason for their choice is that they are well used and understood metrics.

[21] http://www.gate.ac.uk/sale/tao/index.html

It is assumed that their performance scores are indicative of how accurate an automated system is at theme extraction. In other words it is a measure of the ability of the system to extract relevant themes whist constraining the extraction of non-relevant themes.

6.3.3.4. F-measure

The metrics of precision and recall are often equally important yet negatively correlated.

Normally as precision goes up, recall tends to go down and vice versa. For example if a system is conservative then is tends to miss extracting relevant themes (high precision and low recall), and if it is aggressive then it tends to extract non-relevant themes (low precision and high recall).

A balance needs to be made as a system can achieve perfect precision by extracting nothing and so does not generate spurious themes, or it can achieve perfect recall by extracting everything and so not missing any themes.

In Information Retrieval, a method was developed for combining the measures of precision and recall to get a single measure. The F-measure [van Rijsbergen, 1979] (see Equation 6-3) provides a way of combining precision and recall to get a single measure which falls between recall and precision [Chinchor, 1992].

$$\text{F - measure} = \frac{\left(\beta^2 + 1\right) \times \text{P} \times \text{R}}{\beta^2 \times \text{P} + \text{R}}$$

Equation 6-3 – F-measure

Where P is precision, R is recall, and β is the relative importance given to recall over precision. If recall and precision are of equal weight, β = 1.0. For recall half as important as precision, β = 0.5. For recall twice as important as precision, β = 2 .0 [Chinchor, 1992].

The F-measure is higher if the values of precision and recall are similar. So, for β = 1.0, a system which has a precision of 50% and recall of 50% has a higher F-measure than a system which has a precision of 80% and recall of 20% [Chinchor, 1992].

This behaviour demonstrates the importance of an Information Extraction system to be accurate in both precision and recall, and not to focus attempts on just one of the metrics.

The author used a β value of 1.0, as both precision and recall are deemed equally important in an Information Extraction system.

6.4. Human Annotation of Themes

In order to evaluate an Information Extraction system, there needs to be a set of gold documents in which to base a comparison with.

There are two evaluations to be performed. The first evaluates candidate theme extraction, and the second evaluates key theme extraction. For both evaluations a set of gold documents needs to be produced.

6.4.1. Gold Documents

A gold document in Information Extraction is a perfect set of themes that need to be extracted by a system. It is assumed that a gold document is accurate and is flawless.

The actual content of the gold document is dependent on what it is compared against. If it is used to evaluate candidate themes, then the gold document has a list of all candidate themes that a system should extract. Likewise for evaluation of key themes, the gold document contains a list of all key themes that should be extracted.

The generation of a gold document is a manual process. A test document is analysed and its themes are manually extracted. The themes extracted are referred to as gold themes.

This manual theme extraction process is intuitive and there are no rules or restrictions on extraction. There is no limit to the number of themes that can be extracted.

The resultant gold themes are rich, as the human annotators are able to make use of semantic information in the documents.

The annotation of gold corpora can suffer from certain problems. Gold corpora standards can be open to debate as there is usually a lack of agreement on which annotations should be used and how they should be categorised, leading to inconsistencies and potentially errors. Secondly, the annotation of gold corpora is a time intensive task.

6.4.2. Minimising Subjectivity

The main concern when creating a gold document is subjectivity. Unless a document is trivial, it is highly unlikely that multiple annotators produce a set of gold themes that are equivalent.

This does not mean that one annotator is more accurate than another. It just means that the nature of the job is very subjective.

To make the process as scientific as possible, two or more human annotators are used to produce themes for each document. Also all annotators formed a consensus over the questions of 'what constitutes a candidate theme?' and 'what constitutes a key theme?'. The core requirement for the gold themes is that they represent the document, and they can be either a single word or a phrase (see 1.1.2. Automatic Key Theme Extraction for a fuller description).

Each document was themed independently by each annotator. Then each annotator's gold documents were vetted. Then an agreed set of gold themes for each document were chosen.

6.4.3. Candidate Theme Extraction Gold Documents

The gold corpora consists of 6 documents. There were 3 human annotators for this task. Each annotator themed for both candidate themes and also key themes. The results were compared with each other and a gold corpora was agreed.

The aim was for an exhaustive set of candidate themes.

6.4.4. Key Theme Extraction Gold Documents

The gold corpora consists of 117 documents. There were 2 human annotators for this task. Each annotator themed for key themes. The results were compared with each other and a gold corpora was agreed.

The aim was for a set of between 5 – 15 key themes.

It would be preferable to use a larger test set for evaluation, however it was not practical because of the length of time it takes to identify the gold themes and to analyse the results of key theme extraction for each document.

6.5. Active Navigation Theme Extraction

The Active Navigation (AN) theme extraction engine is a commercial product, so detailed assessment of the system cannot be made publicly available as it infringes on Intellectual Property.

An overview of the system is provided. However, the reader must bear in mind that some confidential parts are omitted.

The AN theme extraction engine produces key themes from analysis of document context. The evaluation of this system is black-box, as the actual program being executed is not examined. Only the output is evaluated.

6.5.1. Theme Extraction Architecture

The architecture (see Figure 6-1) is similar to conventional Information Extraction systems. A document is first tokenised and then the tokens are analysed to recognise the language that the document was written in.

If it was written in English then an English-specific stemmer is used to find the root form of the token. The stream of tokens is then used to generate either single-token themes (Token Data) of multi-token themes (Phrase Buffer).

There are a group of files (Language Files) for deciding which themes are selected from a document. These language files contain words and phrases that modify how themes are selected from a document. There are three different categories into which words and phrases can be placed. These are:

- *Noise* – This is a set of words and phrases that appear frequently in the documents and should not appear as themes. They are also used to determine a document's language.
- *Ntheme* – This is a set of words and phrases that should not appear as themes, but which can be used to build a longer phrase. For example, entering 'PC' stops 'PC' from being a theme, but allows phrases such as 'PC World'.

- *Ftheme* – This is a set of words and phrases that you always want to be themes, irrespective of their score.

The final stage is harvesting the themes, where the theming parameters filter the set of themes.

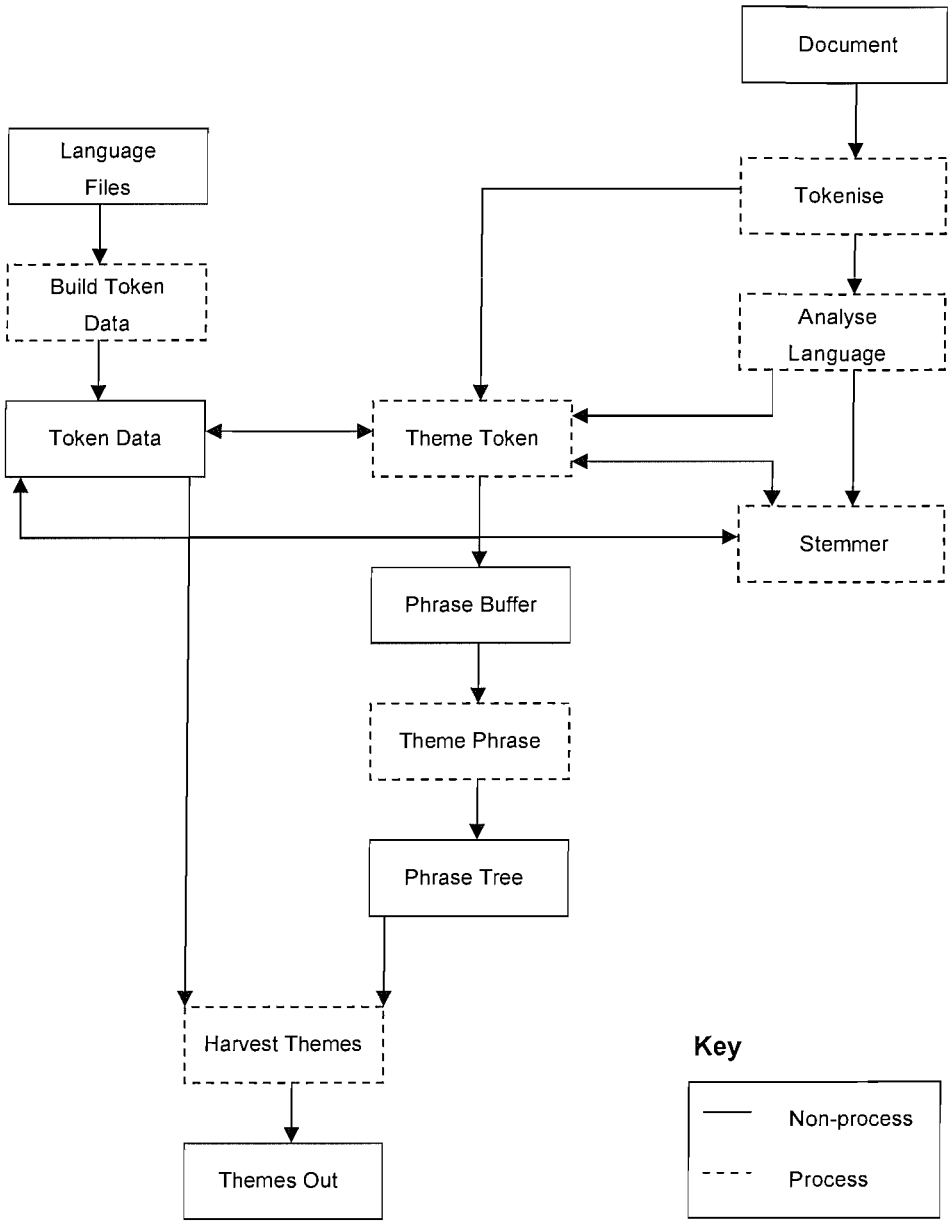

Figure 6-1 – Active Navigation theme extraction architecture

6.5.2. Theming Parameters

The theming parameters (see Table 6-3) have been left at their defaults as they provide good approximations when extracting key themes. The only change was the percentage of themes to use from the document. This was originally 80% and has been increased to 100% to avoid overly aggressive filtering of themes.

Theming Parameter	Setting
Maximum number of themes	20
Percentage of themes to use from the document	100
Minimum score limit	0.02
Phrase bias (percent)	50

Table 6-3 – Theming parameters

In summary, only up to a maximum of 20 key themes that are over the threshold value are output for each test document. Below is a more detailed explanation of the critical parameters.

6.5.2.1. Maximum Number of Themes

This limits the number of themes that are generated for each individual document. This works in addition to the percentage of themes to use from the document to try and ensure that only the most relevant themes are generated by imposing a maximum limit on the number of themes.

6.5.2.2. Percentage of Themes to use from the Document

When each document is themed, a list of all possible words and phrases that are potential themes for that document is stored. This parameter sets the retention level. A percentage of these – the best – are kept the rest are discarded.

6.6. Author's Theme Extraction

The author's system is evaluated against candidate theme extraction and key theme extraction.

When evaluating against candidate theme extraction, all the system's candidate themes for each test document are output.

When evaluating against key theme extraction, the system outputs up to a maximum of 20 key themes that are over a specified score threshold for each test document. This makes the evaluation comparable to the Active Navigation key theme extraction.

6.7. Scoring Strategy

6.7.1. General Points

The scorer takes two documents, one contains the gold themes and the other the system themes. The scorer allocates either a category of correct, partial, missing or spurious to all of the themes in both the documents.

The scoring strategy for candidate themes and key themes are different, this is because the systems for the candidate theme extraction task keep a record of the position information of each extracted theme, whereas the systems for key theme extraction do not.

The consequence is that candidate theme scoring is trivial as the position information allows the scorer to quickly identify the scoring category. For example if a gold theme and the system theme both have the same start and end positions, then the themes must span the same text range and so must be equivalent and therefore categorised as correct. The candidate theme scoring strategy was covered earlier (see 6.3.2. Scoring Strategy).

Without position information, the key theme scorer has to rely on a more sophisticated strategy to best ascertain accurate score category assignments. The key theme scoring is formalised, so that every document in the test sets are scored with consistency. A walkthrough for key theme scoring is also provided.

6.7.2. Scoring Key Theme Extraction

Key theme scoring is case insensitive, this is because a key theme can occur multiple times in a document and its case can be different some of the time, a gold theme and system theme can therefore have different cases, but in essence it is still the same theme.

Below is the list of steps for key theme scoring to be performed in chronological order. Each step ends when no more actions can be taken:

- *Correct* – First match all themes that are exactly the same in both gold and system theme sets. Remove the instance of both from the gold and system theme sets. Increment the correct count for the match.

- *Missing* – Next look for all gold themes that do not occur, even partially, in the system theme set. Remove each missing gold theme. Increment the missing count for each missing gold theme.

- *Spurious* – Next look for all system themes that do not occur, even partially, in the gold theme set. Remove each spurious system theme. Increment the spurious count for each spurious system theme.

  Now only gold and system themes that can partially match with each other remain. Remember that a partial match can subsume more than one theme at a time (see Table 6-2). Partial matching also adheres to a lenient partial matching strategy (see 6.7.3. Lenient Partial Matching).

- *Partial Semantic 1-to-1* – Gold and system themes can be deemed semantically similar even though they are not syntactically equivalent. This occurs when the human annotator and the system both choose themes in the document that are not from the same text span but still represent the same theme.

  This step allows for theme stemming, and reordering of theme phrases, but does not involve synonym matching. For example, 'decommissioning' = 'decommissioned'; 'human cloning' = 'clone humans', but 'science' ≠ 'scientists'.

  This requires high-level semantic matching. This step needs to be performed by a human as an automated scorer cannot provide enough accuracy. This means that key theme scoring is performed manually rather than as an automated process. This step is subjective, though every effort is made to keep the process as consistent as possible.

  This makes the performance scores a more accurate representation of a system's ability to extract key themes from a document, as it takes into account

both syntactic and semantic matching. The scoring does not take into account semantic matching of synonyms as this is outside the scope of the scorer.

If there is gold and system theme that are semantically similar then remove the instance of both from each theme set. Increment the partial count for the partial match.

- *Partial 1-to-1* – If there is a gold and system theme that can only partially match with each other then remove the instance of both from each theme set. Increment the partial count for the partial match.

- *Partial 1-to-M* – If there is a theme (either gold or system) that can only partially match with another theme, regardless if the second theme can ambiguously match with other themes. Then remove the instance of both from each theme sets. Increment the partial count for the partial match.

- *Missing or Spurious* – The remaining themes are either all gold themes or all system themes. These are removed. Increment the missing or spurious counts as appropriate.

6.7.3. Lenient Partial Matching

Partial matching is the most complex case for key theme scoring. In cases where multiple gold themes can match multiple system themes (M:N matches) a lenient matching strategy is used. The aim is to provide as many partial matches as possible.

For example the table below (see Table 6-4) are real sets of key themes produced by human annotation and by Active Navigation over the '409 5k 33' document.

Gold Themes	Active Navigation Themes
Frank H. Netter	Netter collection
Netter's medical illustrations	medical illustrations
13-volume Netter Collection of Medical Illustrations	Netter
Volumes	Volume

Table 6-4 – Lenient partial matching example

There are many ways to score these themes, but by aiming to provide as many partial matches as possible, then these themes give 4 partial matches as shown below:

Partial: '13-volume Netter Collection of Medical Illustrations' → 'Netter collection'
Partial: 'Netter's medical illustrations' → 'medical illustrations'
Partial: 'Frank H. Netter' → 'Netter'
Partial: 'Volumes' → 'Volume'

Rather than for example 2 partial matches and 2 missing as shown below:

Partial: '13-volume Netter Collection of Medical Illustrations' →
'Volume' + 'Netter collection' + 'medical illustrations'
Partial: 'Frank H. Netter' → 'Netter'

Missing: 'Netter's medical illustrations'
Missing: 'Volumes'

The lenient partial matching strategy is used whenever there are potential M:N matches. This provides consistent scoring of these complex cases.

6.7.4. Key Theme Scoring Walkthrough

This shows a demonstration of how the scoring strategy is applied. The table below (see Table 6-5) are real sets of key themes produced by human annotation and by Active Navigation over the 'ACO S Range' document.

Gold Themes	Active Navigation Themes
10 'ACO S Range'	4 'Technical Design Guide'
3 'Specification features'	1 'ACO Technical Design Guide'
12 'Specification of S Range channel units and accessories'	11 'Range channel'
1 'ACO Technical Design Guide'	2 'hydraulic performance'
11 'S Range channel'	10 'ACO'
9 'draining porous asphalt'	9 'asphalt'
2 'Hydraulic performance'	5 'constant'
	6 'drainage'
	9 'porous'
	7 'depth'
	8 'DF'

Table 6-5 – Key theme scoring example

Below (see Table 6-6) is a list of the steps taken, the score category assigned and the themes affected.

Step Number	Step Name	Score Category	Themes Affected
1	Correct	Correct	ACO Technical Design Guide = ACO Technical Design Guide
2	Correct	Correct	Hydraulic performance = hydraulic performance
3	Missing	Missing	Specification features
4	Spurious	Spurious	Technical Design Guide
5	Spurious	Spurious	Constant
6	Spurious	Spurious	Drainage
7	Spurious	Spurious	depth
8	Spurious	Spurious	DF
9	Partial 1-to-1	Partial	draining porous asphalt → porous + asphalt
10	Partial 1-to-M	Partial	ACO S Range → ACO
11	Partial 1-to-M	Partial	S Range channel → Range channel
12	Missing or Spurious	Missing	Specification of S Range channel units and accessories

Table 6-6 – Key theme example results

6.8. Candidate Theme Extraction

6.8.1. Description

This is an evaluation between human candidate theme extraction and the author's system for candidate theme extraction. It is a comparison between the gold candidate themes and the author's candidate themes using the candidate theme scoring strategy. The metrics of precision, recall and F-measure show the system's performance.

6.8.2. Test Set

The test set consists of 6 documents. There are 3 different domains, each has 2 documents. The domains are of BBC news articles, emails and DjVu documents.

The test set provides various challenges as the documents have domain-specific vocabulary; varying levels of grammatical correctness; spelling errors; varying writing styles; varying quantity of content; and can be highly semantic.

The documents were converted to plain-text to provide a fair testing platform.

6.8.3. Results

Document	Correct	Partial	Missing	Spurious	Precision	Recall	F-measure
Email – Minutes	55	51	15	15	84.29%	84.29%	84.29%
Email – US	68	49	6	35	72.55%	93.91%	81.86%
DjVu – ACO	14	23	17	2	92.73%	60.00%	72.86%
DjVu – Anchor	27	13	13	3	91.78%	72.04%	80.72%
BBC – IRA	65	38	12	14	85.71%	87.50%	86.60%
BBC – Iran	51	35	6	14	83.03%	91.95%	87.26%
Averages	47	35	12	14	85.02%	81.62%	82.27%

Table 6-7 – Result of candidate theme extraction

6.8.4. Evaluation of Results

Although the tasks are not equivalent, they are related, so the performance scores for candidate theme extraction have been compared to the MUC-7 Named Entity (NE) in English task (see 6.3.1.3. MUC-7).

My system has an average F-measure of 82.27% which is comparable with the average F-measure of 84.13% of the automated systems for the NE task. What is significant is that the author's system did not use any noun dictionaries so the performance score for candidate theme extraction is noteworthy.

The range of F-measures across the test set is very similar. This is because the author's system used mainly domain independent components so extracts themes regardless of document type.

The average precision and recall scores are similar, but also high (precision 85.02%; recall 81.62%). This means the system has the ability to extract relevant themes whist constraining the extraction of non-relevant themes.

The low recall and F-measure for the 'DjVu – ACO' document can be explained. The author's system uses a shortcut when it recognises that full theme extraction on a sentence cannot be performed because of memory considerations. For these sentences theme extraction is performed on the first parse tree (rather than on all trees). There is therefore a high chance that a full list of candidate themes are not extracted, and consequently be missing from the system's candidate theme set. 11-of-the-17 missing themes in the 'DjVu – ACO' document exist in two sentences that use only the first tree. This shortcut extends to all test documents but is most notable in this document.

It means that recall in documents tend to be lower than what the system can actually achieve.

The average scores for themes that are correct are higher than partially correct themes (correct 47; partial 35). This means that the system is capable of picking out clear boundaries for some themes but not for others.

Analysis of the correct themes indicates that the system is successful at extracting named entities, phrases, and common nouns.

Analysis of the partial themes indicates that the theme boundaries are unclear because of two reasons. The first is due to the choice of modification (pre or post) of themes. For example 'sharp edges' vs. 'injury from sharp edges'; '26 November election date', or '26 November election'. The second reason is the non-trivial task of correctly stipulating and extracting theme phrase boundaries. For example 'suffix DF to the description' vs. 'suffix DF' and 'description'; 'Vienna-based International Atomic Energy Agency (IAEA)' vs. 'International Atomic Energy Agency' and 'IAEA'; 'France, Germany and the UK' vs. 'France' and 'Germany' and 'UK'; 'clarification on traces' vs. 'clarification' and 'traces'.

The average scores for themes that are missing or spurious are similar (missing 12; spurious 14). As stated before this means that the system has struck a balance between extraction of relevant themes whist constraining the extraction of non-relevant themes.

Analysis of the missing themes indicates that the system finds it difficult to extract certain candidate themes. This is due to the choice of the system to extract some types of themes and not others. For example the main missing themes are gerund verbs 'decommissioning', 'branching'; phrases utilising verbs 'twinfish dying', 'encoding issue'; and semantic dates '9/11'. None of these types are considered as themes by the current system.

Analysis of the spurious themes indicates that the system over-generates themes that are either capitalised sentence starters 'Wide', 'Simple'; or common nouns that do not represent candidate themes 'strongest words', 'initial response', 'crunch moment'.

6.9. Key Theme Extraction

6.9.1. Description

Key themes are generated using the candidate theme set. Only the most promising candidate themes become key themes.

This key theme evaluation is a combination of two separate evaluations. The first evaluation is a logical conclusion of the candidate theme results, and is an evaluation between human key theme extraction and the author's system for key theme extraction. It is interesting to see the performance score of the key theme extraction knowing the results of the candidate theme extraction. This is referred to as the small test.

The second evaluation, in comparison, is a large-scale evaluation of key theme extraction. This is an evaluation between human key theme extraction, and both the automated key theme extraction system's of AN, and the author's. This is referred to as the large test.

Both evaluations are comparisons between the gold key themes and the system's key themes using the key theme scoring strategy. The metrics of precision, recall and F-measure show the system's performance.

6.9.2. Test Sets

There are two test sets; one small and one large. The small test set is the same set of documents that were used in candidate theme extraction (see 6.8.2. Test Set).

The large test set consists of 117 documents. They are taken from the Nature Corpus 2001 borrowed from Active Navigation. The documents are from subsets 409 and 410. Each document is approximately one page of text (either 4 or 5 Kb in size), which provides sufficient content for key theme extraction, as well as providing similar size documents for the test set. The documents are from a broad range of topics so are not domain dependent.

The large test set provides a rich document set for evaluation. The documents provide a real test as the documents are by numerous authors, all with unique written styles, and there are many written errors introduced by the authors.

The size of the document set minimises the affect of any human errors made by key theme coring. It also means that the evaluation is not adversely affected by the result of individual documents.

The documents were converted to plain-text to provide a fair testing platform.

6.9.3. Small Test Set Results

Document	Correct	Partial	Missing	Spurious	Precision	Recall	F-measure
Email – Minutes	8	2	1	6	60.00%	90.00%	72.00%
Email – US	9	4	6	5	68.75%	64.71%	66.67%
DjVu – ACO	2	3	2	5	41.18%	63.64%	50.00%
DjVu – Anchor	3	0	4	3	50.00%	42.86%	46.15%
BBC – IRA	7	2	5	5	61.54%	61.54%	61.54%
BBC – Iran	4	5	5	5	56.52%	56.52%	56.52%
Averages	6	3	4	5	56.33%	63.21%	58.81%

Table 6-8 – Result of key theme evaluation – AN (small test set)

Document	Correct	Partial	Missing	Spurious	Precision	Recall	F-measure
Email – Minutes	8	1	2	11	43.59%	80.95%	56.67%
Email – US	9	3	7	8	56.76%	60.00%	58.33%
DjVu – ACO	3	4	0	8	38.46%	100.00%	55.56%
DjVu – Anchor	3	1	3	2	63.64%	53.85%	58.33%
BBC – IRA	6	3	5	11	40.54%	60.00%	48.39%
BBC – Iran	6	3	5	9	45.45%	60.00%	51.72%
Averages	6	3	4	8	48.07%	69.13%	54.83%

Table 6-9 – Result of key theme evaluation – Author (small test set)

6.9.4. Evaluation of Small Test Set Results

The average F-measure's of AN and the authors' systems are (AN 58.81%; Author 54.83%). Over the small test set AN has a performance score that is 4% higher.

The average precision and recall scores show that AN has 8% higher precision but 6% lower recall than the author's system (precision AN 56.33%; Author 48.07%) (recall AN 63.21%; Author 69.13%). This means that AN has a better capability to extract relevant themes whilst the author's system is better at constraining the extraction of non-relevant themes.

The average scores for themes that are correct and partial are exactly the same (correct AN 6; Author 6) (partial AN 3; Author 3). This shows that both systems extract the same quantity of correct and partially correct themes.

The average scores for themes that are missing and spurious are (missing AN 4; Author 4) (spurious AN 5; Author 8). This shows that both systems miss extracting the same quantity of gold themes, but the author's system tends to over-generate themes.

The small test set results are a precursor for the large test set evaluation. This shows a more accurate comparison between AN, and the author's system for key theme extraction.

6.9.5. Large Test Set Results

		Corpora Subset Averages						
Corpora Subset	No. of Docs	Correct	Partial	Missing	Spurious	Precision	Recall	F-measure
409 4k	19	1	3	5	2	67.60%	37.73%	46.62%
409 5k	33	2	3	4	3	50.78%	43.47%	45.65%
410 4k	36	2	2	4	3	54.12%	45.24%	47.76%
410 5k	29	2	2	5	4	50.96%	42.93%	45.03%
	Corpora Average	2	2	4	3	54.58%	42.95%	46.30%

Table 6-10 – Result of key theme evaluation – AN (large test set)

		Corpora Subset Averages						
Corpora Subset	No. of Docs	Correct	Partial	Missing	Spurious	Precision	Recall	F-measure
409 4k	19	2	4	3	3	61.27%	64.55%	60.20%
409 5k	33	4	3	3	5	54.30%	67.95%	59.02%
410 4k	36	3	2	2	4	54.81%	70.24%	59.62%
410 5k	29	3	3	3	6	44.45%	63.74%	51.39%
	Corpora Average	3	3	2	5	53.15%	67.06%	57.51%

Table 6-11 – Result of key theme evaluation – Author (large test set)

6.9.6. Evaluation of Large Test Set Results

The average F-measure's of AN and the authors' systems are (AN 46.30%; Author 57.51%). Over the large test set the author's system has a performance score that is 11% higher (a 24% improvement). This means that the author's system performs better at key theme extraction than that of AN.

The average precision and recall scores show that AN has 1% higher precision but 24% lower recall than the author's system (precision AN 54.58%; Author 53.15%) (recall AN 42.95%; Author 67.06%). This means that both systems have the same capability to extract relevant themes whilst the author's system is considerably better at constraining the extraction of non-relevant themes.

The average scores for themes that are correct and partial are (correct AN 2; Author 3) (partial AN 2; Author 3). This shows that consistently the author's system extracts more correct and partially correct themes than AN.

An analysis shows that AN is better at extracting themes that do not occur frequently in the document. It also shows that AN extracts shorter themes and the author's system has a better capability of extracting longer themes.

The average scores for themes that are missing and spurious are (missing AN 4; Author 2) (spurious AN 3; Author 5). This shows that AN misses extracting more gold themes, but the author's system tends to over-generate themes.

An analysis of the spurious themes generated by the author's system show that many of them are derivations of a matched string. For example, if a full name is used as a gold theme, and if the system extracts both a full name plus a derivation (say, just the surname) then only the full name matches and the derivation becomes spurious.

Also the author's system allows key themes that consist of entirely closed-class words. Typically these never function as key themes. Removal of these themes reduces the numbers of spurious.

6.10. Summary of Results

6.10.1. Summary of Candidate Theme Evaluation

The performance score of the author's system for candidate theme extraction was compared to the MUC-7 Named Entity (NE) in English task (see 6.3.1.3. MUC-7).

The author's system has an average F-measure of 82.27% which is comparable with the average F-measure of 84.13% of the automated systems for the NE task. What is significant is that the author's system does not use any noun dictionaries so the performance score for candidate theme extraction is noteworthy.

6.10.2. Summary of Key Theme Evaluation

The author's system and AN were both evaluated for key theme extraction. There were two tests, one small and one large.

For the small test, the average F-measure's of AN and the authors' systems are (AN 58.81%; Author 54.83%). Over the small test set AN has a performance score that is 4% higher.

For the large test, the average F-measure's of AN and the author's system are (AN 46.30%; Author 57.51%). Over the large test set the author's system has a performance score that is 11% higher (a 24% improvement).

The large test provides a more accurate measure of system performance. The results show that the author's system performs better at key theme extraction than that of AN.

6.11. Summary of Chapter

This chapter has provided an evaluation of the performance between the author's and Active Navigation's theme extraction.

The chapter began with a history of Information Extraction evaluation and provided the scoring strategy and metrics for which the evaluations were based. The performances were measured using established metrics for precision, recall and F-measure. The performances that were evaluated were the ability of the system to extract candidate themes from a plain-text document, and also the ability to extract key themes from a plain-text document.

The candidate theme evaluation showed that the author's system has the ability to extract relevant themes whist constraining the extraction of non-relevant themes.

The key theme evaluation showed that the author's system performed better at key theme extraction than that of Active Navigation. For the large test set, the average F-measure's of Active Navigation and the author's system were (Active Navigation 46.30%; Author 57.51%). Over the large test set the author's system had a performance score that was 11% higher (a 24% improvement).

7. Conclusion

This chapter provides the conclusions for the system. It provides a summary of the chapters in the thesis, a discussion on whether the system has met the research objectives, and how the research can be used to improve Active Navigation. The final section outlines future work, this includes how to improve key theme extraction, how these key themes can be applied, and looking forward to their use in the Semantic Web.

7.1. Summary of Previous Chapters

Chapter 1 provided an introduction to information management, the problem with current approaches, and suggested how automatic key theme extraction can benefit an information management system. The chapter also included the non-triviality of written information in the form of natural language texts used in these systems. The chapter concluded with a definition of the research objectives for this project including the motivation, goals and deliverables.

Chapter 2 provided the aims of Active Navigation and the importance that themes play in achieving those aims. The chapter included an overview of Active Navigation, an outline of its branding, navigation solution and competitors. The chapter set the importance of extracting key themes from a document by providing an in-depth overview of a themes purpose, theories, a typical theme extraction process, and methods to rank extracted themes.

Chapter 3 provided a background history of Natural Language Processing (NLP), in particular a detailed discussion into the two polarised language models of constituency and dependency on which all Information Extraction systems are based. This was followed by an overview of the parsers available for both language models, and concluded with a comparison of benchmark systems used in NLP.

Chapter 4 provided an analysis of the issues of constituency and dependency language models. The main issue found with both language models was that of ambiguity. In the constituency model the ambiguity flowed through the whole length of the pipeline affecting each component. In the dependency model the main ambiguity stemmed from

the use of constituency components to build dependency structures. Both language models are strongly affected by the choice of grammar as it affected the mapping of a language.

Chapter 5 provided the justification for the choices of the automatic key theme extraction system. In particular the need for a new lossless architecture and the benefits it achieves. This required a new formalism for representing information which was called a quad. An architecture was researched by the author to the extent of fully understanding the components required for theme extraction. Fundamental to this architecture was an ethos for lossless information. To preserve the lossless ethos the components were built from scratch as often benchmark systems perform pruning on the output. To this end each component produced and preserved information. Then the issue of computational feasibility of a lossless architecture was raised. The chapter concluded with an overview of the system implementation.

Chapter 6 provided an evaluation of the performance between the author's and Active Navigation's theme extraction. The chapter began with a history of Information Extraction evaluation and provided the scoring strategy and metrics for which the evaluations were based. The performances were measured using established metrics for precision, recall and F-measure. The performances that were evaluated were the ability of the system to extract candidate themes from a plain-text document, and also the ability to extract key themes from a plain-text document. The candidate theme evaluation showed that the author's system has the ability to extract relevant themes whist constraining the extraction of non-relevant themes. The key theme evaluation showed that the author's system performed better at key theme extraction than that of Active Navigation. For the large test set, the average F-measure's of Active Navigation and the author's system were (Active Navigation 46.30%; Author 57.51%). Over the large test set the author's system had a performance score that was 11% higher (a 24% improvement). The chapter concluded with a summary of the results.

7.2. Research Objectives

This research has presented the topic of automatic key theme extraction as a method for information management, specifically the extraction of pertinent information from natural language texts.

7.2.1. Motivation

The motivation for this research was to achieve improved accuracy in automatic key theme extraction in natural language texts. The performance was evaluated against an industrial context, which was provided by Active Navigation Ltd, a content management system.

7.2.1.1. Key Theme Extraction

The system extracted key themes from each document. These themes represented the key content of the document. The extraction process automated a human ability to extract key themes.

7.2.1.2. Natural Language Texts

Natural language texts were used to evaluate the extraction process as they provided a real document set that is used in industry. These documents were unstructured and only contained plaintext to prevent metadata information from being gleaned from them. Successful key theme extraction of these non-trivial texts is beneficial to other types of text as the extraction process did not require any extra information apart from the plaintext. The biggest challenge of these documents was interpreting the meaning of a document and extracting only key themes from it because of the complexity of natural language, in particular its ambiguous nature.

7.2.1.3. Key Theme Evaluation

The key theme evaluation showed that the author's system performed better at key theme extraction than that of Active Navigation. For the large test set, the average F-measure's

of Active Navigation and the author's system were (Active Navigation 46.30%; Author 57.51%). Over the large test set the author's system had a performance score that was 11% higher (a 24% improvement).

7.2.2. Goals

The goals that needed to be achieved were to produce a useable grammar and language model that can be implemented into Active Navigation's current software. It was assumed that the research extended the current software's capabilities by providing a more accurate navigation experience of documents.

7.2.2.1. Natural Language Grammar

A natural language grammar was produced. This was the most important component of the system as it formalised how a language was used. The grammar provided a formal description of natural language using syntactic and linguistic information. As well as syntactic information the grammar also provided syntactic functions to supply extra richness such as subject and object.

The grammar was allowed to be ambiguous so that the expressiveness of natural language was mapped. This ambiguity kept the lossless information ethos as all possibilities are explored rather than a simple deterministic choice.

The grammar provided a large coverage of natural language. It catered for both frequently and non-frequently occurring structures.

The grammar was used in the system to produce syntactic parse trees from which the candidate themes were extracted.

Active Navigation currently do not use a grammar, it is beneficial to see the affect of the application of such a linguistic tool on the quality of its key theme extraction.

7.2.2.2. Language Model

A new language model was produced. The constituency and dependency language models were analysed for strengths, weaknesses and issues, the result of the analysis was the formation of a new language model, which put at its foundation the use of Natural Language Processing techniques to improve theme extraction.

The constituency language model highlighted the weaknesses of sub-standard components used in the pipeline, the loss of critical information, and error propagation.

The dependency language model highlighted that constituency components were used to build dependency grammars, and that frequently the dependency grammars compromised the model by not using a full lexicon, as creating the knowledge is a laborious task.

To resolve these issues the new language model was built from scratch and based on a constituency model, so that the boundaries and the work required of each component were known, and the flow of information from one component to the next was well understood. With all components their goals, input and output were clearly defined. The specific details for the new language model were defined in the deliverables.

7.2.3. Deliverables

The deliverables required were an architecture for theme extraction using a pipeline of individual processing components that adhered to a lossless information strategy. The aim of the architecture was to accurately extract key themes from natural language documents using a fusion of linguistic, statistical and grammar rules.

7.2.3.1. Lossless Architecture

Traditional language models use deterministic methods to make decisions along the pipeline. The problem was decision-making components usually lead to component filtering which propagated a loss of information. Also many systems re-use benchmark systems that may not be the most accurate, such sub-standard components introduced

errors. The combination of losing information and error propagation created a deterioration of the quality of information that was propagated by the architecture.

The new language model was built based on the idea of lossless information. This was the retention of a complete information history for all the components in the architecture. The aim of the architecture was to keep all the decisions, and pass them on through other components, so that each component has all the information at hand and can make a more accurate evaluation of all the alternatives. A lossless architecture stopped the pitfalls of component filtering.

Information persistence in the architecture preserved a history of rule selection. This improved the design of the rule interactions as the affect of each rule became transparent, and any underperforming rules that were identified were modified for effectiveness. This enforced transparency in the architecture and minimised error propagation by reduction of the error passed from one component to another.

The lossless architecture kept track of all choices using quads. A quad was a novel way of representing a slice of information and a tool that enforced the underlying concept of lossless information.

7.2.3.2. Architecture Components

The architecture was built modularly. This was so that each component had distinct input and output boundaries and had unique task descriptions. The modularisation in essence provided a balance of work for each component. It also allowed for more transparent component interaction.

The components in the architecture were knowledge engineered, developed using experience of language and made use of human intuition. The components were rule and statistical based.

To preserve the lossless ethos the components were built from scratch as often benchmark systems perform pruning on the output. To this end each component produced and preserved information.

The further down the component pipeline the larger the contexts that the components worked with. The early components worked on single or multiple tokens whilst the latter took sentence and document contexts. This was because latter components handled more complex processes and also used the larger contexts to provide better disambiguation.

7.3. Improvements for Active Navigation

The author has designed, implemented and evaluated a new language model based on an architecture of individual components that adhered to a lossless ethos. This lossless architecture has shown that it was capable of providing a higher accuracy of extracting key themes from natural language texts than that of Active Navigation (AN).

The key theme evaluation showed that both systems had a similar capability to extract relevant themes (precision AN 54.58%; Author 53.15%) whilst the author's system was considerably better at constraining the extraction of non-relevant themes (recall AN 42.95%; Author 67.06%), a significant 56% improvement in recall.

The evaluation did show that AN was better at extracting themes that occur infrequently in a document, and that AN extracts shorter themes whereas the author's system had a better capability of extracting longer themes.

It has been shown that a lossless architecture using predominantly linguistic components can yield more accurate results for key theme extraction. The lossless architecture and the components, especially the grammar can be utilised by AN, in particular to increase the constraining power over non-relevant themes and to improve on the extraction of longer themes.

7.4. Future Work

This section concludes the thesis by highlighting future work that is aimed at improving key theme extraction, discussing the application of these extracted themes, and looking forward to their use in the Semantic Web.

7.4.1. Improving Key Theme Extraction

There are various factors that influenced the result of the evaluations. The main ones are the use of natural language texts, the omission of semantic components, the natural language grammar, the dependency of key and candidate themes, the scoring of key themes and computational feasibility. These factors are discussed below.

7.4.1.1. Natural Language Texts

The test sets used in the evaluations were all plain-text natural language documents. Plain-text was a non-trivial problem because there was a lack of information mark-up in a document. Which meant that the determination of themes was more difficult, because there are no visual cues (e.g. for emphasis) or discourse information in the document (e.g. headings).

As well as the plain-text issue, the natural language context provided many other complex problems such as different written styles, use of grammar and ambiguities. The combination of these difficulties made theme extraction more difficult.

It was assumed that the quality of results for key theme extraction was related to the type of document used and that plain-text natural language documents are particularly difficult. Simpler document types are assumed to yield improved results.

7.4.1.2. Semantics

The system did not include semantic components, so there were no semantic lexicons for word sense disambiguation or providing semantic information. The content word

dictionary used in the system only contains lexical entries, and there was no noun dictionary for providing semantics.

It would be interesting to see what affect semantic components have on resolving ambiguities and key theme extraction.

7.4.1.3. Natural Language Grammar

The grammar dictated which structures are extracted as themes, because of this constraint there were potential types of themes which were not extracted. For example the main missing themes were gerund verbs 'decommissioning', 'branching'; phrases utilising verbs 'twinfish dying', 'encoding issue'; and semantic dates '9/11'. None of these types were considered as themes by the system.

The grammar extracts incorrect themes because they were labelled as themes in the syntactic parse trees. For example the system over-generated themes that were either capitalised sentence starters 'Wide', 'Simple'; or common nouns that did not represent candidate themes 'strongest words', 'initial response', 'crunch moment'.

The grammar also provided unclear theme boundaries because of two reasons. The first was due to the choice of modification (pre or post) of themes. For example 'sharp edges' vs. 'injury from sharp edges'. The second reason was the non-trivial task of correctly stipulating and extracting theme phrase boundaries. For example 'suffix DF to the description' vs. 'suffix DF' and 'description'.

The natural language grammar was a major component in the system for extrapolating theme information, but more techniques are required to intelligently choose the correct theme representation given the information in the syntactic parse trees.

7.4.1.4. Dependency of Key and Candidate Themes

In the author's system, key theme extraction was entirely dependent on candidate theme extraction. Key themes were generated from analysing the candidates and choosing the most pertinent ones. If the candidate was not extracted from the original document then it never became a key theme.

This dependency strongly biased the results, and in future it might be preferable to perform the key theme extraction directly over the document, rather than the candidate themes, which might provide an improvement in results.

7.4.1.5. Key Theme Scoring

Key theme scoring can be improved by identifying important themes that either explicitly exist or can be inferred from the candidate theme set. In technical terms this requires theme stemming, reordering of theme phrases, and synonym matching. All of which, are absent from the current system. For example, if a candidate theme set contains themes for 'teaching of evolution' and 'evolution teaching'; or 'tobacco firm' and 'tobacco company'; then the system must realise that they are semantically the same so the scoring must take into account that they are the same and not separate themes.

The scoring strategy should also take into account other factors that can influence the score of the theme (see 2.2.3. Ranking Themes) as currently the scoring strategy was affected by a small set of factors which can bias the score.

7.4.1.6. Computational Feasibility

A lossless architecture was able to improve the accuracy of theme extraction whilst minimising the propagation of error. However, this preservation of information had a detrimental affect on computational feasibility as all permutations were tried rather than a best-first approach.

As mentioned before this research was aimed at quality rather than speed, but some memory problems manifested into the system. The author had to make a decision to speed up certain components so that the system does not run out of memory. Unfortunately the cost of these shortcuts was that the ethos of the lossless architecture was not kept which meant that there was a reduction in the potential power of some components and ultimately vital information was lost.

These memory considerations meant that the power of the lossless architecture was not fully utilised as only a limited set of options were pursued. Trying more options can produce different sets of results, and perhaps more accurate results.

7.4.2. The Application of Key Themes

The accurate extraction of key themes was essential as they can be used as a solid base for other Active Navigation information navigation tasks. These include advanced search, categorisation, building summaries, finding related documents, and dynamic linking. Improving these navigation techniques increases the effectiveness of the content management system.

Take categorisation and dynamic linking for example, after extracting the key themes from a document the categorisation task becomes more accurate in populating a taxonomy and at the same time reduces human intervention. With dynamic linking, improved links can be dynamically generated between related documents based on their themes. This is based on the assumption that the processes of theme extraction and linking are symmetrical, so that improved theme extraction ultimately improves the quality of the linking.

In conclusion, the extraction of key content benefits an organisation by assisting it to manage and optimise its data-repositories for efficient use. This is a critical process for any organisation that wants to avoid being overwhelmed by an explosion in the volume of information.

A consequence of this is that human intervention is minimised and should result in a reduction in the quantity of manual labour and ultimately cost.

7.4.3. The Semantic Web

The research for this project has focussed primarily on the role of Natural Language Processing techniques to deliver an Information Extraction solution for natural language texts. However, a fast emerging factor is the need for an organisation to be able to capitalise on technology advancements such as the Semantic Web, which utilise richer representations of information rather than just natural language texts. This requires a development of new methods to extract information.

The Semantic Web vision, as articulated by Tim Berners-Lee, is of a Web in which resources are accessible not only to humans, but also to automated processes

[Bechhofer et al., 2002]. The key idea is to have data on the web defined and linked in such a way that its meaning is explicitly interpretable by software processes rather than just being implicitly interpretable by humans. The consequence is that the Semantic Web becomes dependant on the production of machine-definable and machine-understandable information.

To realise this vision, it is necessary to associate metadata with Web resources. One mechanism for associating such metadata is annotation. In particular, the annotation of resources with semantic metadata that provides some indication of the content of a resource. This metadata is often used as part of an ontology. An ontology is a conceptual model that contains all the relevant concepts and their relationships and rules that are applicable to a specific domain. The problem with metadata annotation and building ontologies is that the information is painstaking to create.

A system that exploits the use of semantic information for information navigation is Conceptual Open Hypermedia Services Environment (COHSE) [Carr et al., 2001] and [Bechhofer et al., 2002]. COHSE combined the Distributed Link Service (see 2.1.2.2. Dynamic Links) with an ontology service to form a conceptual hypermedia system to enable documents to be linked using metadata describing their contents. Traditional link services provide destination anchors based only on the keywords found in the source document, however, COHSE links on concepts in the ontology found in the document. The structure made explicit in an ontology enabled COHSE to also broaden and narrow the destinations of a link.

COHSE and other systems seeking to take advantage of a Semantic Web require more semantic annotation of current Web resources. Millard [Millard et al., 2003] suggests that Information Extraction is one of a number of promising methods for enriching Web documents with semantics for the purpose of future semantic interpretation.

It is a future goal of the author to see how the key theme extraction system can be modified to extract metadata, in particular concepts and relationships, to see if can be used to automatically populate an ontology. This may be a difficult transition as the construction of ontologies from text can suffer from data sparseness if the key concepts are implicit or never stated in the text.

7.4.4. Statistical Methods

This research has emphasised the importance of linguistic tools to promote theme extraction. However there is a great deal of literature using statistical methods to perform similar tasks. It would be wise to explore statistical systems to compare with the author's theme extraction. For example the Minimum Description Length (see 3.3.3. The Rationale for Parsing).

Statistical systems can use the regularities found in language to form rules for finding important themes. They require less linguistic knowledge and can provide results in a shorter period of time compared with the author's current system.

7.4.5. Alternate Evaluations

The current evaluation only compares the output of Active Navigation and the author's theme extraction systems with gold standard documents. This only gives a figure of how effective a system is at presenting the key themes in a document. However, it does not provide a conclusive evaluation of the value of the themes extracted and how effective these themes are at retrieving related documents.

It would be worthwhile to consider alternate evaluations. For example it would be valuable to use client surveys to assess evaluation documents. Clients of Active Navigation could be used to assess the documents retrieved for particular themes; this would provide a real-world evaluation of theme extraction. This would demonstrate the effectiveness of the system to retrieve only relevant documents based on themes, and more generally the navigation of documents based on themes.

Another form of evaluation may involve a knowledge acquisition study in which language experts familiar with Information Retrieval participate in assessing evaluation documents. This sort of study may result in the production of a reliable and agreed upon standards for themes and document navigation.

8. Bibliography

[Abney, 1989] Abney, S. (1989). "*A Computational Model of Human Parsing*". Journal of Psycholinguistic Research, 18 pp 129-144.

[Aho et al., 1986] Aho, A, V., Sethi, R., and Ullman, J. (1986). "*Compilers: Principles, Techniques and Tools*". Addison-Wesley, Reading, MA, Second Edition, 1986.

[Ahrenberg et al., 1990] Ahrenberg, L., Jönsson, A., and Dahlbäck, N. (1990). "*Discourse Representation and Discourse Management for a Natural Language Dialogue System*". In Proceedings of the 2nd Nordic Conference on Text Comprehension in Man and Machine, Täby, Stockholm, 1990.

[Appel, 1998] Appel, A, W. (1998). "*Modern Compiler Implementation in Java*". Cambridge University Press; ISBN: 0521583888.

[Aretoulaki, 1996] Aretoulaki, M. (1996). "*COSY-MATS: A Hybrid Connectionist-Symbolic Approach to the Pragmatic Analysis of Texts for their Automatic Summarisation*". Ph.D. Thesis, Dept. of Language Engineering, University of Manchester Institute of Science and Technology (UMIST), Manchester, U.K., March.

[Aretoulaki, 1997] Aretoulaki, M. (1997). "*COSY-MATS: An Intelligent and Scalable Summarisation Shell*". In Proceedings of the ACL'97 / EACL'97 Workshop on Intelligent Scalable Text Summarisation (ISTS'97), Madrid, Spain, July 1997.

[Aretoulaki et al., 1998] Aretoulaki, M., Scheler, G., and Brauer, W. (1998). "*Connectionist Modeling of Human Event Memorization Processes with Application to Automatic Text Summarization*". In Proceedings

of AAAI Spring Symposium on Intelligent Text Summarization, Stanford University, California, March, 1998.

[Arnola, 1998] Arnola, H. (1998). "*On Parsing Binary Dependency Structures Deterministically in Linear Time*". In Kahane & Polguere (eds), Workshop on Dependency-Based Grammars, COLING-ACL 1998, pp 68-77, Montreal.

[Aycock & Horspool, 1999] Aycock, J., and Horspool, N, R. (1999). "*Faster Generalized LR Parsing*". In Proceedings of the 8th International Conference on Compiler Construction, pp 32-46, 1999.

[Aycock & Horspool, 2001] Aycock, J., and Horspool, N, R. (2001). "*Directly-Executable Earley Parsing*". In Proceedings of the 10th International Conference on Compiler Construction, pp 229-243, 2001.

[Aycock & Horspool, 2002] Aycock, J., and Horspool, N, R. (2002). "*Practical Earley Parsing*". The Computer Journal, 45(6): pp 620-630, 2002.

[Aycock et al., 2001] Aycock, J., Horspool, N, R., Janoušek, J., and Melichar, B. (2001). "*Even Faster Generalized LR Parsing*". Acta Informatica, Volume 37, Issue 9, pp 633-651, 2001.

[Baldwin & Villavicencio, 2002] Baldwin, T., and Villavicencio, A. (2002). "*Extracting the Unextractable: A Case Study on Verb-Particles*". In Proceedings of the 6th Conference on Natural Language Learning (CoNLL-2002), Taipei, Taiwan, 2002.

[Bechhofer et al., 2002] Bechhofer, S., Carr, L., Goble, C., Kampa, S., and Miles-Board, T. (2002). "*The Semantics of Semantic Annotation*". ODBASE: First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems 2519, Irvine, California. 2002.

[Berger & Printz, 1998a] Berger, A., and Printz, H. (1998). "*A Comparison of Criteria for Maximum Entropy / Minimum Divergence Feature Selection*". In Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP) 1998, pp 97-106, Granada, Spain.

[Berger & Printz, 1998b] Berger, A., and Printz, H. (1998). "R*ecognition Performance of a Large-Scale Dependency-Grammar Language Model*". In International Conference on Spoken Language Processing (ICSLP 1998), Sydney, Australia.

[Berrut & Palmer, 1986] Berrut, C., and Palmer, P. (1986). "*Solving Grammatical Ambiguities within a Surface Syntactical Parser for Automatic Indexing*". In ACM-SIGIR, Pisa, September.

[Biber et al., 1999] Biber, D., Johansson, S., Leech, G., Conrad, S., and Finegan, E. (1999). "*Longman Grammar of Spoken and Written English*". Longman.

[Bies et al., 1995] Bies, A., Ferguson, M., Katz, K., and MacIntyre, R. (1995). "*Bracketing Guidelines for Treebank II Style Penn Treebank Project*". University of Pennsylvania, 1995.

[Blache, 2000] Blache, P. (2000). "*Constraints, Linguistic Theories and Natural Language Processing*". In Natural Language Processing, Christodoulakis, D. (ed), Lecture Notes in Artificial Intelligence 1835, Springer-Verlag.

[van den Brand et al., 2002] van den Brand, M, G, J., Scheerder, J., Vinju, J., and Visser, E. (2002)."*Disambiguation Filters for Scannerless Generalized LR Parsers*". In Horspool, N. (ed), Compiler Construction, Volume 2304 of Lecture Notes in Computer Science, pp 143-158, Grenoble, France, April 2002. Springer-Verlag.

[Brill, 1994] Brill, E. (1994). "*Some Advances in Transformation-Based Part of Speech Tagging*". In Proceedings of the 12th National Conference on Artificial Intelligence, 1994.

[Brill & Resnik, 1994] Brill, E., and Resnik, P. (1994). "*A Rule- Based Approach to Prepositional Phrase Attachment Disambiguation*". In 15th International Conference on Computational Linguistics (COLING) 1994.

[Carbonell, 1983] Carbonell, J, G. (1983). "*Discourse Pragmatics and Ellipsis Resolution in Task-Oriented Natural Language Interfaces*". In Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL-83): pp 164-168.

[Cardie, 1993] Cardie, C. (1993). "*A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis*". In Proceedings of the Eleventh National Conference on Artificial Intelligence, pp 798-803, Washington, DC. AAAI Press / MIT Press.

[Carr et al., 1995] Carr, L., De Roure, D., Hall, W., and Hill, G. (1995). "*The Distributed Link Service: A Tool for Publishers, Authors and Readers*". Proceedings of Fourth International World Wide Web Conference: The Web Revolution, Boston, Massachusetts, USA, December 1995.

[Carr et al., 1998] Carr, L., De Roure, D., Davis, H., and Hall, W. (1998) "*Implementing an Open Link Service for the World Wide Web*". World Wide Web Journal 1(2).

[Carr et al., 2001] Carr, L., Bechhofer, S., Goble, C., and Hall, W. (2001). "*Conceptual Linking: Ontology-based Open Hypermedia*". Tenth World Wide Web Conference, Hong Kong. 2001.

[Carroll & Bricoe, 2001] Carroll, J., and Bricoe, T. (2001). "*High Precision Extraction of Grammatical Relations*". In Proceedings of IWPT 2001, pp 78-89.

[Carroll & Charniak, 1992] Carroll, G., and Charniak, E. (1992). "*Two Experiments on Learning Probabilistic Dependency Grammars from Corpora*". In AAAI 1992 Workshop Program: Statistically-Based NLP Techniques, San Jose, California.

[Chan et al., 2000] Chan, S., Lai, T., Gao, W., and T'sou, B. (2000). "*Mining Discourse Markers for Chinese Textual Summarization*". In Udo Hahn, Lin, C., Mani, I., and Radev, D, R. (eds), Proceedings of the Workshop on Automatic Summarization at the 6th Applied Natural Language Processing Conference and the 1st Conference of the North American Chapter of the Association for Computational Linguistics, Seattle, WA, April 2000.

[Charniak, 2001] Charniak, E. (2001). "*Immediate Head Parsing for Language Models*". In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, 2001.

[Chelba et al., 1997] Chelba, C., Engle, D., Jelinek, F., Jimenez, V., Khudanpur, S., Mangu, L., Printz, H., Ristad, E., Rosenfeld, R., Stolcke, A., and Wu, D. (1997). "*Structure and Performance of a Dependency Language Model*". In Proceedings of Eurospeech, Volume 5, pp 2775-2778, Rhodes, Greece, 1997.

[Chiaramella et al., 1986] Chiaramella, Y., Defude, D., and Bruandet, M. (1986). "*IOTA: A Full Text Information Retrieval System*". In Proceedings of the 9th ACM-SIGIR, Pisa, Italy, pp 207-213.

[Chinchor, 1992] Chinchor, N, A. (1992). "*MUC-4 Evaluation Metrics*". In Proceedings of the Fourth Message

Understanding Conference, McLean, Virginia, pp 22-29, June 16-18, 1992.

[Chinchor, 1999] Chinchor, N, A. (1999). "*Overview of MUC-7 / MET-2*". In Proceedings of the Message Understanding Conference MUC-7.

[Ciravegna & Lavelli, 1999] Ciravegna, F., and Lavelli, A. (1999). "*Full Text Parsing using Cascades of Rules: An Information Extraction Perspective*". In Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics, Bergen, Norway, 1999.

[Collins & Brooks, 1995] Collins, M., and Brooks, J. (1995). "*Prepositional Phrase Attachment through a Backed Off Model*". ACL 3rd Workshop on Very Large Corpora, pp 27-38, Cambridge, Massachusetts, June 1995.

[Covington, 1990] Covington, M, A. (1990). "*A Dependency Parser for Variable-Word-Order Languages*". In Brown, H, U. (ed), Computer Assisted Analysis and Modelling on the IBM 3090, Cambridge, MA, MIT Press.

[Covington, 1994] Covington, M, A. (1994). "*Discontinuous Dependency Parsing of Free and Fixed Word Order*". Work in Progress. Research Report AI-1994-02, University of Georgia, Athens, Georgia 30602 USA, April 1994.

[Covington, 2000] Covington, M, A. (2000). "*A Fundamental Algorithm for Dependency Parsing*".

[Cunningham et al., 1999] Cunningham, H., Gaizauskas, R., Humphreys, K., and Wilks, Y. (1999). "*Experience with a Language Engineering Architecture: Three Years of GATE*". Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP, 1999.

[Cunningham et al., 2002a] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). "*GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications*". Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia, US.

[Cunningham et al., 2002b] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., and Dimitrov, M. (2002). "*Developing Language Processing Components with GATE (User's Guide)*". Technical Report, University of Sheffield, UK, (2002).

[Davis et al., 1992a] Davis, H., Hall, W., Heath, I., Hill, G., and Wilkins, R. (1992). "*Microcosm: An Open Hypermedia Environment for Information Integration*". Technical Report, Southampton University, 1992. CSTR 92-15.

[Davis et al., 1992b] Davis, H., Hall, W., Heath, I., Hill, G., and Wilkins, R. (1992). "*Towards an Integrated Information Environment with Open Hypermedia Systems*". Proceedings of the Fourth ACM Conference on Hypertext (ECHT '92), Lucarella, D., Nanard, J., Nanard, M., and Paolini, P. (eds), Milan, Italy, pp 181-190, December 1992.

[Davis et al., 1993] Davis, H., Hutchings, G., and Hall, W. (1993). "*Microcosm: A Hypermedia Platform for the Delivery of Learning Materials*". Technical Report, Southampton University, 1993. CSTR 93-10.

[Douthat, 1998] Douthat, A. (1998). "*The Message Understanding Conference Scoring Software User's Manual*". In 7th Message Understanding Conference Proceedings, MUC-7, 1998.

[El-Beltagy et al., 1999] El-Beltagy, S., De Roure, D., and Hall, W. (1999). "*A Multiagent System for Navigation*

Assistance and Information Finding". In Proceedings of the Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp 281-295.

[Elworthy, 1995] Elworthy, D. (1995). "*Tagset Design and Inflected Languages*". In Proceedings of the ACL SIGDAT Workshop, Dublin, 1995.

[Elworthy, 1999] Elworthy, D. (1999). "*A Finite-State Parser with Dependency Structure Output*".

[Fellbaum, 1998] Fellbaum, C. (1998). "*Towards a Representation of Idioms in WordNet*". In Harabagiu, S. (ed), Proceedings of the Workshop on Usage of WordNet in Natural Language Processing. Systems, COLING/ACL 1998, Montreal, Canada.

[Grice, 1971] Grice, H, P. (1971). "*Meaning*". In Steinberg, D, D., and Jakobovits, L, A. (eds), Semantics. An Interdisciplinary Reader in Philosophy, Linguistics and Psychology, pp 53-59. Cambridge University Press.

[Grosz & Sidner, 1986] Grosz, B, J., and Sidner, C, L. (1986). "*Attention, Intentions, and the Structure of Discourse*". Computational Linguistics, 12(3) pp 175-204, July-September.

[Grünwald, 2004] Grünwald, P, D. (2004). "*Tutorial on Minimum Description Length*". Chapter in Grünwald, P, D., Myung, I, J., and Pitt, M, A. (eds), Advances in Minimum Description Length: Theory and Applications, MIT Press.

[Hajicova & Sgall, 1984] Hajicova, E., and Sgall, P. (1984). "*From Topic and Focus of a Sentence to Linking in a Text*". In Bara, B, G., and Guida, G. (eds), Computational Models of Natural Language Processing, pp 151-163. Elsevier Science Publishers B. V. (North Holland).

[Hearst & Plaunt, 1993] Hearst, M, A., and Plaunt, C. (1993). "*Subtopic Structuring for Full-Length Document Access*". In Proceedings of the 16th ACM-SIGIR, Pittsburgh, PA, pp 59-69.

[Hirschberg & Litman, 1993] Hirschberg, J., and Litman, D. (1993). "*Empirical Studies on the Disambiguation of Cue Phrases*". Computational Linguistics, 19(3) pp 501-530.

[Horspool & McLean, 1996] Horspool, N, R., and McLean, P. (1996). "*A Faster Earley Parser*". In Proceedings of the 6th International Conference on Compiler Construction, pp 281-293, 1996.

[Ifill, 2002] Ifill, T. (2002). "*Seeking the Nature of Idioms: A Study in Idiomatic Structure*". Thesis.

[Infante-Lopez et al., 2002] Infante-Lopez, G, G., Rijke, M., and Sima'an, K. (2002). "*A General Probabilistic Model for Dependency Parsing*". In Proceedings of the BNAIC 2002, Leuven, Belgium.

[Inui et al., 1997] Inui, K., Sornlertlamvanich, V., Tanaka, H., and Tokunaga, T. (1997). "*A New Formalization of Probabilistic GLR Parsing*". In Proceedings of the International Workshop on Parsing Technologies, 1997.

[Järvinen & Tapanainen, 1998] Järvinen, T., and Tapanainen, P. (1998). "*Towards an Implementable Dependency Grammar*". In Processing of Dependency-Based Grammars, Montreal, Canada. Kahane, S., and Polguere, A. (eds), COLING-ACL 1998, Association for Computational Linguistics, pp 1-10, Universite de Montreal.

[Krymolowski & Dagan, 2002] Krymolowski, Y., and Dagan, I. (2002). "*Targeted Partial Dependency Parsing*".

[Kübler & Hinrichs, 2001] Kübler, S., and Hinrichs, E, W. (2001). "*From Chunks to Function Argument Structure: A Similarity Based Approach*". Association for Computational Linguistic, 39th Annual Meeting

and 10th Conference of the European Chapter, Proceedings of the Conference, July 9-11, 2001, Toulouse, France. Morgan Kaufmann Publishers, 2001, pp 338-345.

[Lavie & Tomita, 1993] Lavie, A. and Tomita, M. (1993). "*GLR* - An Efficient Noise-skipping Parsing Algorithm for Context Free Grammars*". In Proceedings of Third International Workshop on Parsing Technology, pp 123-134, 1993.

[Lin & Pantel, 2001] Lin, D., and Pantel, P. (2001). "*Discovery of Inference Rules for Question Answering*". Natural Language Engineering 7(4) pp 343-360, 2001.

[Marino et al., 1987] Marino, M., Spiezio, A., Ferrari, G., and Prodanof, I. (1987). "*An Efficient Context-Free Parser for Augmented Phrase-Structure Grammars*". In Proceedings of the Third Conference on European Chapter of the Association for Computational Linguistics, Copenhagen, Denmark, pp 196-202, 1987.

[Maynard et al., 2000] Maynard, D., Cunningham, H., Bontcheva, K., Catizone, R., Demetriou, G., Gaizauskas, R., Hamza, O., Hepple, M., Herring, P., Mitchell, B., Oakes, M., Peters, W., Setzer, A., Stevenson, M., Tablan, V., Ursu, C., and Wilks, Y. (2000). "*A Survey of Uses of GATE*". Technical Report CS-00-06, University of Sheffield, Department of Computer Science, 2000.

[Maynard et al., 2001] Maynard, D., Tablan, V., Ursu, C., Cunningham, H., and Wilks, Y. (2001). "*Named Entity Recognition from Diverse Text Types*". In Recent Advances in Natural Language Processing 2001 Conference, Tzigov Chark, Bulgaria, 2001.

[McPeak & Necula, 2004] McPeak, S., and Necula, G, C. (2004). "*Elkhound: A Fast, Practical GLR Parser*

Generator". Compiler Construction, Proceedings. 2985, pp 73-88.

[Melby, 1987] Melby, A, K. (1987). "*Statutory Analysis*". Technical Report, Birgham Young University, Provo, Utah, October.

[Mellish, 2004] Mellish, C. (2004). "*Resolving Structural Ambiguity in Generated Speech*". Belz, A., Evans, R., and Piwek, P. (eds), Natural Language Generation, Third International Conference, INLG 2004, Brockenhurst, UK, July 2004.

[Menzel & Schröder, 1990] Menzel, W., and Schröder, I. (1990)."*Decision Procedures for Dependency Parsing using Graded Constraints*". In Proceedings of ACL 1990.

[Middleton et al., 2001] Middleton, S., DeRoure, D., and Shadbolt, N. (2001). "*Capturing Knowledge of User Preferences: Ontologies in Recommender Systems".* In Proceedings of the ACM K-CAP'Oi, 2001, Victoria, Canada: ACM Press.

[Millard et al., 2003] Millard, D, E., Alani, H., Kim, S., Weal, M, J., Lewis, P., Hall, W., De Roure, D., and Shadbolt, N, R. (2003). "*Generating Adaptive Hypertext Content from the Semantic Web*". In Proceedings of 1st International Workshop on Hypermedia and the Semantic Web, Nottingham, UK.

[Miller et al., 1990] Miller, G, A., Beckwith, R, T., Fellbaum, C, D., Gross, D., and Miller, K, J. (1990). "*WordNet: An On-line Lexical Database.*" International Journal of Lexicography, 3(4), pp 235-244.

[Nakata et al., 1998] Nakata, K., Voss, A., Juhnke, M., and Kreifelts, T. (1998). "*Collaborative Concept Extraction from Documents*". In Proceedings of the 2nd Int. Conf. on Practical Aspects of Knowledge

Management (PAKM 98), Basel, Switzerland, 29-30.

[Nakatani, 1991] Nakatani, C, H. (1991). "*Resolving a Pragmatic Prepositional Phrase Attachment Ambiguity*". 29th Annual Meeting of the Association for Computational Linguistics, 18-21 June 1991, University of California, Berkeley, California, USA, Proceedings. ACL 1991 pp 351-352.

[Nivre, 2002] Nivre, J. (2002). "*Two Models of Stochastic Dependency Grammar*". MSI Report 02118. Växjö University, School of Mathematics and Systems Engineering.

[Nivre, 2003] Nivre, J. (2003). "*Optimizing a Deterministic Dependency Parser for Unrestricted Swedish Text*". In Proceedings of Promote IT, Gotland University, 3-5 May 2003.

[Nivre & Nilsson, 2003] Nivre, J., and Nilsson, J. (2003). "*Three Algorithms for Deterministic Dependency Parsing*". NODALIDA and Related Activities, Rejkyavik, May 2003.

[Numazaki & Tanaka, 1990] Numazaki, H., and Tanaka, H. (1990). "*A New Parallel Algorithm for Generalized LR Parsing*". In Proceedings of the 13th International Conference on Computational Linguistics - Volume 2, Helsinki, Finland, pp 305-310, 1990.

[Oflazer, 1999] Oflazer, K. (1999). "*Dependency Parsing with an Extended Finite State Approach*". In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, College Park, MD.

[O'Hara, 2002] O'Hara, K. (2002). "*The Internet: A Tool for Democratic Pluralism?*". Science as Culture 11(2): pp 287-298.

[Pantel & Lin, 2000] Pantel, P., and Lin, D. (2000). "*An Unsupervised Approach to Prepositional Phrase Attachment*

using Contextually Similar Words". In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, pp 101-108, Hong Kong.

[Pantel & Lin, 2002] Pantel, P., and Lin, D. (2002). "*Discovering Word Senses from Text*". In ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, May 2002.

[Paradis & Berrut, 1996] Paradis, F., and Berrut, C. (1996). "*Experiments with Theme Extraction in Explanatory Texts*". In Second International Conference on Conceptions of Library and Information Science, CoLIS 2. pp 433-437.

[Paskin, 2001] Paskin, M, A. (2001). "*Grammatical Bigrams*".

[Porter, 1980] Porter, M. (1980). "*An Algorithm for Suffix Stripping*". Program (Automated Library and Information Systems) 14 (3): pp 130-7, July 1980.

[Quirk et al., 1985] Quirk, R., Greenbaum, S., Leech, G., and Svartvik, J. (1985). "*A Comprehensive Grammar of the English Language*". Longman.

[Ratnaparkhi, 1998] Ratnaparkhi, A. (1998). "*Statistical Models for Unsupervised Prepositional Phrase Attachment*". In Proceedings of the 36th ACL and 17th COLING, pp 1079-1085.

[Ratnaparkhi et al., 1994] Ratnaparkhi, A., Reynar, J., and Roukos, S. (1994). "*A Maximum Entropy Model for Prepositional Phrase Attachment*". In Proceedings of the ARPA Workshop on Human Language Technology, pp 250-255, Plainsborg, N J, March 1994.

[Rissanen, 1978] Rissanen, J. (1978). "*Modeling by Shortest Data Description*". Automatica, Vol. 14, pp 465-471.

[van Riemsdijk & Williams, 1986] van Riemsdijk, H., and Williams, E. (1986). "*Introduction to the Theory of Grammars*". Cambridge: The MIT Press.

[van Rijsbergen, 1979] van Rijsbergen, C, J. (1979). "*Information Retrieval*". Butterworths, London, 2nd edition.

[Small, 1980] Small, S, L. (1980). "*Word Expert Parsing: A Theory of Distributed Word-based Natural Language Understanding*". Doctoral Dissertation, Department of Computer Science, University of Maryland, September 1980.

[Strzalkowski et al., 2000] Strzalkowski, T., Stein, G, C., Wise, G, B., and Bagga, A. (2000). "*Towards the Next Generation Information Retrieval*". In Proceedings of the RIAO-2000, 6th International Conference on Intelligent Multimedia, Information Retrieval Systems and Management, Paris, April, 2000.

[Tapanainen & Järvinen, 1997] Tapanainen, P., and Järvinen, T. (1997). "*A Non-Projective Dependency Parser*". In Proceedings of the 5th Conference on Applied Natural Language Processing, Association for Computational Linguistics, Washington, D.C.

[Thorup, 1994] Thorup, M. (1994). "*Controlled Grammatic Ambiguity*". ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 16, Issue 3 (May 1994), pp 1024-1050, 1994.

[Tomita, 1990] Tomita, M. (1990). "*The Generalized LR Parser / Compiler v8-4: A Software Package for Practical NL Projects*". In Proceedings of the 13th International Conference on Computational Linguistics - Volume 1, Helsinki, Finland, pp 59-63, 1990.

[Wagner & Graham, 1997] Wagner, T, A., and Graham, S, L. (1997). "*Incremental Analysis of Real Programming Languages*". In Proceedings of the ACM SIGPLAN 1997 Conference on Programming

Language Design and Implementation, Las Vegas, Nevada, United States, pp 31-43, 1997.

[Woods, 1997] Woods, W, A. (1997). "*Conceptual Indexing: A Better Way to Organize Knowledge*". A Sun Labs Technical Report: TR-97-61. Editor, Technical Reports, 901 San Antonio Road, Palo Alto, California 94303, USA.

[Xia & Palmer, 2001] Xia, F., and Palmer, M. (2001). "*Converting Dependency Structures to Phrase Structures*". In the Proceedings of the Human Language Technology Conference (HLT 2001), San Diego, CA, March 18-21, 2001.

[Zhou, 2002] Zhou, M. (2002). "*A Block-Based Robust Dependency Parser for Unrestricted Chinese Text*". The second Chinese Language Processing Workshop attached to ACL 2000, Hong Kong, October 8th, 2000.