

Abstract

Text Data Mining (TDM) is a nascent field falling under the umbrella of Business Intelligence (BI). Some of the data vital for a business's success will only be available in free text format. TDM addresses the major challenge of incorporating this data into pre-existing internal data warehouses and decision systems. We will deal in this report with the theoretical techniques that could have application in TDM and we will attempt to identify research opportunities that arise from application of currently under-utilised techniques.

Contents

1	Introduction	4
2	Characters	5
2.1	Regular Expressions	6
2.2	Searching Marked-up Text	7
3	Words	9
3.1	Word Frequency	9
3.2	Morphology	10
3.3	Parts of Speech	10
4	Sentences	12
4.1	Grammars	12
4.2	Parsing	15
4.2.1	Parsing PCFGs	16
4.2.2	Markov grammars	17
4.2.3	Lexicalised Parsing	18
4.2.4	Transformation-based Parsing	19
4.3	Data sets	20
4.3.1	The Penn treebank	20
4.3.2	The SUSANNE corpus	20
5	Meaning	21
5.1	Collocations	21
5.2	Latent Semantic Analysis	22
5.3	Logical Forms	25
5.3.1	Definite-clause grammars	27
5.3.2	Semantics as DCG Augmentations	27
5.3.3	Quasi-Logical Form	28
6	Context	30
6.1	Converting from quasi-logical form to logical form	31
6.2	Reference Resolution	32
6.3	Rhetorical Structure Theory	34
7	Research Directions	36
7.1	Information Integration	36
7.2	Active Summaries	37
7.3	Research Projects	38

List of Figures

1	Finite State Machine representation of ‘a[bc]*d’	6
2	An example of eXtensible Markup Language	8
3	A simple context-free grammar	13
4	Combining operations for TAGs (after [13])	14
5	A parse tree for the sentence <i>The cat sat on the mat.</i>	15
6	The word by context matrix for our example.	24
7	The reconstructed word by context matrix for our example <i>after</i> dimensional reduction.	29
8	A grammar specifying both syntax and semantics.	29
9	Title and Abstract from an article from the October 1972 edi- tion of <i>Scientific American</i>	34
10	RST analysis of Lactose example	35

List of Tables

1	Equivalent Regular Expressions and FSMs corresponding to the rules for regular languages.	7
2	Most common bigram collocations in the 1990 <i>New York Times</i> corpus	22
3	Most common bigram collocations in the 1990 <i>New York Times</i> corpus filtered by their part of speech	23
4	Data for toy example: Title of technical memos; c = human- computer interaction, g = graph theory	24
5	Three contextually dependent sentences and their QLFs. . . .	32
6	Examples of RST coherence relations	35

1 Introduction

Text Data Mining (TDM) is a nascent field falling under the umbrella of Business Intelligence (BI). While every consultant and company markets their particular technologies and solutions for dealing with data as “business intelligence”, a consensus is emerging that reporting and analysis of business data beyond that required for day-to-day operations is safely the province of BI systems. These systems are required to support non-routine decisions and detect new problems.

Straight data mining also known, more informatively, as knowledge discovery in databases (KDD) is an aspect of BI that is better developed. KDD applications discover trends and patterns across very large datasets in a (semi-)automatic manner. The data is usually collected from different areas of the business and stored in large systems of relational databases called data warehouses.

However, there is a great deal of business data that is not neatly contained in the fields of a relational database. Some data vital for a business’s success will only be available in free text format:

- electronic mail from customers containing feedback.
- internal documents; memos, policies and presentations.
- technical reports on new technology.
- annual and other statutory reports.
- Print media; newspapers, magazines and news wires.

Increasing this information is available internally via an intranet or externally on the World Wide Web, where most of the information is in hypertext markup language (HTML) format. A major challenge addressed by TDM is to incorporate this data into the internal data warehouses and decision systems.

Some idea of the benefits that could accrue to business if sophisticated TDM applications were available can be seen from the following highly selective list of possible application areas.

stock market sector analysis - issue buy, sell and hold notices for sectors based on news feeds and company reports

interest rates predict interest rate rises and falls based on statements from central banks and other economic data

health automatically assess advisability of particular treatments from literature searches

human resources matching job seekers to positions

journalist's workbench scan newswires, the WWW and archives for information on a story. Automatic personalisation of responses based on style of article required (feature, news, business, human interest *etc.*)

human-computer interaction accessing your computer using natural language

In this context “sophisticated” methods scale appropriately and handle uncertainty robustly.

There is already a scholarly field of endeavour engaged in the practice of text data mining, corpus-based computational linguistics. There is also intersection with natural language processing, a sub-field of artificial intelligence [26].

In this survey we will attempt to survey the extensive literature available from both of these areas with relevance to TDM. We will concentrate on techniques from all of these fields that deal with textual rather than spoken language. Starting at the simplest level with individual characters and moving through words to sentences. We then look for meaning in these syntactic structures and finish with higher-level structures within discussions. For each of these we will attempt to provide a careful description of the methods used, an example of it use and one or more hopefully useful, but possibly not canonical, references.

The last section will attempt to identify research opportunities that arise from application of currently unutilised techniques. These opportunities will be assessed against their technical innovation, applicability, whether a path to application exists and alignment with the BI group's research plan.

2 Characters

It is possible to think of a document as just a long string of characters, punctuation and whitespace, without identifying larger structures like words, sentence or paragraphs. One of the simplest operations that can be envisaged on this structure is the free text search. Regular expressions are now standard technology for searching and we cover them and their relationship to finite state automata and regular languages before discussing the subtleties of searching in markup languages that now dominate.

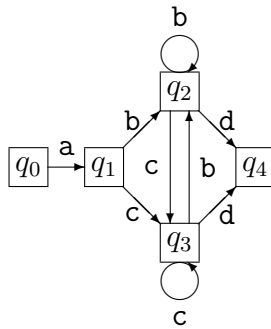


Figure 1: Finite State Machine representation of ‘a[bc]*d’

2.1 Regular Expressions

Regular expressions are a standard methods for specifying strings. Almost every text processing program uses a similar language; Perl, Emacs and even Microsoft Word use them to specify search patterns. They were initially developed by Kleene [15] and their use in text searching was described by Thompson [32].

Regular expressions can be represented as strings or as finite state machines (FSM). For example, the pattern “an a followed by any number of bs and cs and ending with a d” can be written as ‘a[bc]*d’ or as the FSM of Figure 1.

A finite state machine consists of a set of states, the q_i , and transitions between the states, here labelled by the letters in the pattern. It begins operation in the initial state, q_1 and moves to the other states as it encounters the letters labelling the transitions.

The set of all text strings that match a particular regular expressions constructed from an alphabet, Σ , is a called a regular language. A regular language is a type of formal language, of which more in the section on grammar (section 4.1). For now we will list the axioms that define the class of regular languages:

1. The empty set, \emptyset , is a regular language.
2. For every element, a of the alphabet, $\{a\}$ is a regular language
3. If L_1 and L_2 are regular languages then

$$L_1 \cdot L_2 = \{xy|x \in L_1, y \in l_2\}$$

is a regular language.


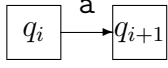
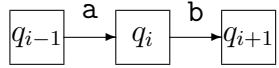
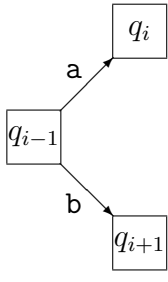
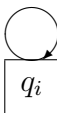
Rule	Regular Expression	Finite State Machine
1.	The empty string $\epsilon = ''$	
2.	'a'	
3.	'ab'	
4.	'[ab]'	
5.	'a*'	

Table 1: Equivalent Regular Expressions and FSMs corresponding to the rules for regular languages.

4. If L_1 and L_2 are regular languages then

$$L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$$

is a regular language.

5. $L_1^* = \{x^n \mid x \in L_1, n \in \mathbf{N}\}$, the Kleene closure of L_1 , is also a regular language.

Each of the axioms has an associated regular expression form and FSM state transition which we show in Table 1

2.2 Searching Marked-up Text

From the simplified description of FSMs above we can see how they can be used to identify if its input is the pattern in question. If it reaches it's final state, q_4 in our example, then we have a match, but if it fails, receives a d in state q_2 for example, then there is no match.

```

<recordset total="9">
  <row id="1">
    <field name="EmployeeID">1</field>
    <field name="LastName">Davolio</field>
    <field name="FirstName">Nancy</field>
    <field name="Title">Sales Representative</field>
    <field name="TitleOfCourtesy">Ms.</field>
    <field name="HireDate">5/1/92</field>
    <field name="City">Seattle</field>
  </row>
</recordset>

```

Figure 2: An example of eXtensible Markup Language

Two questions arise as the search proceeds through the document

When do we signal a match? Searching for ‘a[bc]*’ and finding “abcd” which of “a”, “ab”, or “abc”, all legal matches, do we return

Where do we start searching again? If we report “abc” as the match from our previous example, do we start again at “b” or “d”.

The usual answers to the first of these questions is encapsulated in the phrase “left-most longest match”. In practice this means we signal a match as early as possible in our document and match as much as possible. To quote the POSIX standard

The search is performed as if all the possible suffixes of the string were tested for the prefix matching the pattern.

The search is resumed after the end of the match that has been signalled.

This seems eminently sensible until we consider Figure 2 which shows some text that has been “marked up” using the Extensible Markup Language (XML) to represent data in an employee database. If we ask “How do I find all the employees in Seattle?”, the problem with the left-most longest regime is that the obvious regular expression

```

<row .*> .* <field name="City">Seattle</field> .* </row>

```

matches from the first occurrence of a Seattle employee to the end of the last row. We have used . as the usual abbreviation for a regular expression that matches any character in the alphabet.

It would be possible to solve this problem by designing a more sophisticated regular expression but to do this for every possible markup language and search seems wasteful. Clarke and Cormack [10] suggest changing the search paradigm to “shortest non-nested” as a way of short-circuiting this problem. With this interpretation we signal the smallest match that is not wholly included inside another possible match. With this change our naïve regular expression now returns the correct span of text.

3 Words

A more complicated structure apparent in text is the word and if we view our document as a list of words a richer set of structures and possible data analyses present themselves. It is at this level that corpus-based computational linguistic methods become relevant. We will look, in turn, at word frequency, morphology and parts of speech.

3.1 Word Frequency

Word frequency counts, while simplistic, show how quickly the complications of natural language processing arise. An analysis of word frequencies of the novel *Tom Sawyer* in [21] shows that nearly 50% of the words occur only once and 90% of words occur less than 10 times.

Because all corpuses are finite, we have a serious case of the “sparse data problem”, as there will be words in our test corpus that we will not have met in training. We will have to assign these words a notional zero probability. This is a problem as maximum-likelihood techniques are not robust to small probabilities. Approaches to estimating more realistic values for the probabilities of unseen words are called *smoothing* or *discounting*. We will discuss two discounting methods, Witten-Bell and Good-Turing.

It is a tautology to say that a zero frequency word is one we have yet to see, and when we see it we will see it for the first time, but this is the observation that underlies Witten-Bell smoothing. This technique models the probability of unseen words using the probability of those that appear just once. If there are T unique words in a collection of N in total, then the probability of a new word is $T/(N + T)$. If there are Z unseen words in the test corpus and we divide this probability equally amongst them then the probability, p_i of the i -th word is

$$p_i = \begin{cases} \frac{T}{Z(N+T)} & i|c_i = 0 \\ \frac{c_i}{N+T} & i|c_i \neq 0 \end{cases}$$

where c_i is the frequency (possibly zero) of the i -th word.

Rather than reweighting all words equally, another approach is to take into account the number of words, N_c , having frequency, c . Using Good-Turing smoothing the new counts, c^* are

$$c^* = (c + 1) \frac{N_{C+1}}{N_C}$$

Using this method it is usual to threshold this process and leave counts above a certain level as they are. Also low counts (*i.e.* one) are often treated as zero, along the lines of the previous paragraph.

3.2 Morphology

Words also have internal structures that are not strictly related to the characters that they consist of. The study of these structures is called morphology and there are three types

inflectional a word is derived from a root word acquiring additional features but maintaining it's function (*e.g.* dog \rightarrow dogs)

derivational where the function of a word is changed by the addition of some structure (*e.g.* able \rightarrow ably)

compounding independent words are joined to form a new one (*e.g.* button + hole \rightarrow buttonhole)

If we are to use words to signal the presence of significant concepts in our text it will be important not to be distracted by these variations. Porter [24] describes a process for stripping words to their roots using transformation rules (but see [16] for caveats).

3.3 Parts of Speech

Each word also has a grammatical function called its *part-of-speech*. These include nouns (**N**), verbs (**V**), adjectives (**ADJ**), adverbs (**ADV**), articles (**ART**). One of the first natural-language processing tasks to receive a proper statistical treatment was part-of-speech tagging (*c.f.* [7, 8].) Unfortunately a word does not necessarily have a single part of speech. Consider the following example:

The	can	will	rust
article	modal-verb	modal-verb	noun
	noun	noun	verb
	verb	verb	

Under each word is some of its possible parts of speech in order of frequency, with the correct tag emboldened.

Our document, $D = (w_n)_{n=1}^N$ is a sequence of words, w_n (punctuation counts as a word). We have a list of the parts of speech of our language, $P = (t_i)_{i=1}^M$ and a training corpus $T = (w_j, t_j)_{j=1}^L$, which is a list of words and their associated tags. The task is to assign the correct tags to the words in the document, D . There are several ways this can be done.

Maximum Likelihood taggers We calculate the probability, $p(t|w)$ that each word, w has tag t by counting co-occurrences in our training corpus. Then for each word, w_n , in our document that appears in T we simply assign the tag, t that maximises $p(t|w_n)$. If the word doesn't appear in our corpus then we tag it as a proper noun. This corresponds, in our example above, to assigning to each word the part of speech immediately below it. Despite our gloomy results in this case this process is usually about 90% accurate for less pathological sentences.

Hidden Markov taggers If in addition we collect statistics from our corpus on the probability, $p(t_i|t_{i-1})$ of tag t_i following tag t_{i-1} and assign the tag that maximizes $p(w_n|t)p(t|t_{n-1})$. For instance while “can” is most often a **modal-verb**, this part of speech is almost never preceded by an **article** making the **article+noun** pairing much more likely. This process of taking into account the tag of the previous word boosts success to about 95%.

Transformational taggers After using one of the previous taggers we can see the errors it has made and would like to make some changes to fix them. In the context of the “rusty can” example above we might like a rule that says “change **modal-verb** after **article** to **noun**.” This is the basis of transformation-based taggers (*c.f.* Brill [5]).

We begin by choosing a large set of rules from which to select useful examples from (*e.g.* all rules of the form “change **tag₁** to **tag₂** after **tag₃**”). Then we tag the training corpus with one of the statistical taggers above and then select the rule that corrects the most mistakes. We continue by adding those rules to our list that correct the most mistakes that remain after the application of all the previously selected rules, until adding further rules has little effect.

When applied to test sentences these taggers have similar success rates to the best statistical taggers, but, if the rules are turned into a finite state automaton which is then compiled, they can function up to an order of magnitude faster.

Human taggers are consistent with each other about 98% of the time giving an upper bound to expected performance of any automatic part-of-speech tagger.

4 Sentences

If we now consider our document broken up in to sentences then we can begin to investigate the syntax of our documents. The Shorter Oxford English Dictionary [18] defines syntax as

the arrangement of words by which their connexion and relation in a sentence are shown.

After discussing grammars, we will look at methods for analysing free text in terms of these grammars and at available data sets.

4.1 Grammars

Our ideas about syntax are usually captured in a *grammar*. The grammar of the English language is usually described in terms of it's phase structure with a phase associated with each of the major parts of speech (*i.e.* sentences (S), noun phrases (NP), verb phrases (VP), prepositional phrases (PP) *etc.*) A grammar is a concise way of specifying the relations between phrases that are acceptable in a language. They usually consist of

- a set of terminal symbols (the words of the languages)
- a set of non-terminals (the phrases of the language)
- an initial non-terminal (usually the largest structure of interest)
- a set of rules that defines the relations between the terminal and non-terminal symbols

A grammar has a degree of expressiveness termed its *generative power*. Chomsky [9] arranges languages in a hierarchy of increasing generative power based on the structure allowed in the rewrite rules of the grammar. His four types are

(left) regular grammars For this type of grammar the only rules allowed are those that map a single non-terminal to a single terminal symbol optionally followed by another non-terminal. They are equivalent in power to finite-state automata

context-free grammars In this class of grammars we lift the restriction on the structure of the right-hand side of the rules allowing mappings from a single non terminal to any combination of terminal and non-terminal symbols. This is by far the most important class of grammars as they are simple enough to allow efficient algorithms to be constructed around them, but expressive enough to capture some of the complexities of natural language.

context-sensitive grammars We now permit any number of symbols on the left-hand side of the rules, allowing the introduction of a context in which the rule can be applied. We require, however, that there be at least as many symbols on the right-hand side.

recursively enumerable grammars These grammars have no restrictions on the forms their rewrite rules can take. They are equivalent to Turing machines in their expressive power.

In Figure 3 we show a simple context-free grammar (CFG) for a subset of English. We have left out the *lexicon*, those rules that state explicitly what words are a noun or verb *etc.*

- | | |
|---------------------------|-----------------------------|
| 1. $S \rightarrow NP VP$ | 5. $VP \rightarrow V$ |
| 2. $NP \rightarrow ART N$ | 5. $VP \rightarrow V NP$ |
| 3. $NP \rightarrow N$ | 7. $VP \rightarrow V NP PP$ |
| 4. $PP \rightarrow P NP$ | 8. $VP \rightarrow V PP$ |

Figure 3: A simple context-free grammar

Context-free grammars, while sophisticated enough for computer programming languages, are not considered sophisticated enough to capture the linguistic complexity of a natural language[29]. In order to overcome this difficulty other more sophisticated grammars (*i.e.* those which have a higher generative power) are used.

The most immediately appealing are the tree-adjoining grammars (TAGs) [13], which prescribe rewrite rules for trees whose leaf nodes are the terminal symbols of our language rather than the string of words itself (see section 4.2 for reasons why this might be useful). Figure 4 illustrates the two important operations defined of TAGS; adjoining and substitution. A wide coverage TAG for English is part of the XTAG system [34] together with a parser and a morphological analyser. Other grammars that are extensively studied include Head Phrase Structure Grammars [27], Linear Indexed Grammars,

Figure 4: Combining operations for TAGs (after [13])

and Combinatory Categorical Grammars. Rather helpfully it is proved in [33] that these grammars are all of equal generative power.

4.2 Parsing

Parsing is the process of linking the part-of-speech tags into a tree structure that indicates the grammatical structure of the sentence with the interior nodes represent phrases, links represent the application of grammatical rules and leaf nodes represent words. See Figure 5 for an example. Parse trees are usually drawn with the analysed examples written out horizontally and constituency structure indicated by slanted lines above. For typographical convenience we have used a different (but isomorphic) convention, which displays the example vertically, with a parsable unit on each line.

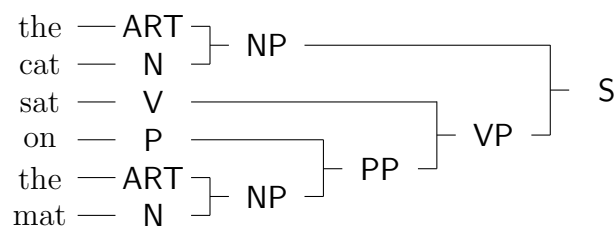


Figure 5: A parse tree for the sentence *The cat sat on the mat.*

A great deal of work has been done on parsing the formal languages described in the previous section, mostly by computer scientists interested in compilers. The definitive reference from this point of view is [2] although a more practical exposition can also be found in [1]. Parsing is essentially a search through the *parse forest*, the set of all possible parse trees and there are, superficially, two classes top-down and bottom-up, although the more successful ones use techniques from both classes.

Top-down or goal-directed parsers build from the initial non-terminal down to the leaves (left to right in figure 5), backtracking when the leaves generated fail to match those in the sentence that we are attempting to parse. *Bottom-up* or data-directed parsers start with the words and try to build a tree with the initial non-terminal at the top (right to left in figure 5), backtracking if there is no rule that can combine the non-terminals constructed to date.

While these methods, and their more sophisticated brethren, work very

well on formal programming languages, natural languages pose problems that simply do not arise in that context. Consideration of the sentence

The salespeople sold the dog biscuits.

illustrates the case where there are two grammatically acceptable parses of a particular sentence. The usual approach is to construct all acceptable parses and then go through a process of *disambiguation* to select the correct one. In the statistical context this separation seems artificial as the concept of the Probabilistic Context-Free Grammar (PCFG) shows.

4.2.1 Parsing PCFGs

A *Probabilistic Context-Free Grammar* (PCFG) is a context-free grammar which has a probability associated with each rule normalised so that the probabilities of the rules associated with a particular non-terminal sum to 1.

Disambiguation is achieved by selecting the parse tree with the highest probability. The probability of a parse tree, π for a sentence, S is given by

$$P(S, \pi) = \prod_{n \in \pi} P(r(n)) \quad (1)$$

where $p(r(c))$ is probability of the rule, r that has been applied to expand non-terminal, n in the parse tree.

The *probabilistic CYK (Cocke-Younger-Kasami)* algorithm [2] is a bottom-up recursive parser that is the next level of sophistication above the “enumerate all parse trees and pick the one with the highest probability according to (1)” algorithm. We assume the words, w_i of our sentence, $S = (w_i)_{i=1}^N$ has been tagged with their parts of speech, t_i .

The base case is where a non-terminal, A , expands to a single part-of-speech, t_i . The probability of this is just $P(A \rightarrow t_i)$.

The inductive step deals with strings of tags, $(t_l)_{l=i}^j$ of length at least two. A non-terminal, A gives rise to such a string if there are other non-terminals, B and C , for which $A \rightarrow BC$, $B \rightarrow (t_l)_{l=i}^k$ and $C \rightarrow (t_l)_{l=k+1}^j$ are all rules in the grammar. The probability of this step is the product of the probabilities of the three rules. Induction gives us the last two probabilities while the first is an explicit rule in the grammar. If there is more than one applicable rule we choose the one that gives the highest probability. The parse tree can be recovered by remembering the rules with highest probability that were applied at each step.

The CYK algorithm and, indeed, any other dynamic programming approaches to parsing PCFGs (*e.g.* minimum edit distance, Viterbi, Earley) suffer as effective parsers due to the fundamental independence assumption that the grammars require. They are unable to model structural or lexical dependencies that arise in natural language.

A *structural dependency* arises when the probability of a rule changes depending on its position in a sentence. An example of this is the placement of pronouns in declarative sentences. Jurafsky and Martin [14, p. 456] quote a study that shows that pronouns were three times more likely to be used as the subject of a sentence than the object. This kind of information cannot be encoded in the single rule that inserts a pronoun into a sentence, **NounPhrase** \rightarrow **Pronoun**.

Lexical dependencies can be illustrated by the sentence

Moscow sent more than 100,000 soldiers into Afghanistan.

The problem here is to attach the prepositional phrase “into Afghanistan” to the verb phrase “sent ...” or the noun phrase “more than 100,000 soldiers”. In a PCFG the decision as to which attachment to prefer can only be made once and must be the same for all verbs. In our example the verb “sent” needs a destination which a prepositional phrase starting “into ..” is perfectly placed to provide. A grammar that maintained preferences for each lexically distinct verb would be able to give the correct parse.

The approaches taken to remedy these deficiencies are many and varied. We will consider two, Markov grammars and lexicalization of parsers.

4.2.2 Markov grammars

Magermann [19] considered Markov grammars as a way of encoding structural dependencies in a model of natural language. Although he uses the language of decision trees, the terminology that we use here, suggested by Charniak [8], is quite natural given the forms these grammars take.

Instead of maintaining a probability for each of a list explicit rules, we calculate probabilities of a component, c_i , (terminal or non-terminal) appearing in the expansion of a non-terminal, p , based on previous component in the expansion, c_{i-1}

$$P(c_i|p, c_{i-1}).$$

If there is no previous component then we write $P(c_1|p, c_0)$ for the probability that this component initiates an expansion. We have added a null component, c_0 .

To encode the structural difference noted earlier we could require that

$$P(\text{Pronoun}|\text{VerbPhrase}, \text{Verb}) < P(\text{NounPhrase}|\text{VerbPhrase}, \text{Verb}).$$

Notice that we are no longer limited by fixed rules motivated by strict grammatical structure in this approach. To link up with the previous section we can calculate the probability of a “rule”, r for non-terminal, p , with components c_i is then

$$P(r|p) = \prod_{c_i \in r} p(c_i|p, c_{i-1})$$

In spite of this increased flexibility these simple Markov grammars are only marginally better than the more straightforward PCFGs. In practise they are usually used in a lexicalised form, where we use the word themselves and not just their parts of speech as input to the parser.

4.2.3 Lexicalised Parsing

We can’t simply collect statistics on the grammatical use of individual words as this leaves us with an even sparser data set than for part-of-speech tagging. One way of minimising the number of combinations is to consider only the the *head* of a each phrase, which is the most important lexical item. Several modern linguistic theories of syntax include this notion [27]. A couple of examples illustrates the fairly natural nature of this concept.

The head of a noun phrase is obviously its main noun; the head of “more than 100,000 soldiers” being “soldiers”. Prepositional phrases are headed by the preposition as exemplified by the phrase “**into** Afghanistan.” The thinking behind this approach is that even if the exact phrase doesn’t appear in the training set, the words in the rest of the phrase may be associated separately with head in other contexts.

Lexicalised parsers collect two sorts of statistics. Firstly, the probability, $P(r|h)$, of which rule, r , should be applied given the head, h of the phrase, p to be expanded. Secondly, the probability that a sub-phrase, q , has head, h , given the head of the phrase being expanded, m (for “mother”). The total probability for a parse, π , of a Sentence, S is then

$$p(S, \pi) = \prod_{c \in \pi} P(h(c)|m(c))P(r(c)|h(c))$$

Using this information we can solve our prepositional attachment problem. Even if the application of both rules is equally likely

$$P(\text{VP} \rightarrow \text{V NP PP} | \text{“sent”}) = P(\text{VP} \rightarrow \text{V NP} | \text{“sent”}).$$

We can distinguish the correct parse by the preference that “into” shows for binding to “sent” rather than “soldiers”,

$$P(\text{“into”}|\text{“sent”}) > P(\text{“into”}|\text{“soldiers”}).$$

One way of thinking about lexicalized parsers is to imagine them as CFGs with a profusion of rules, one for each word in the vocabulary. While this may seem impractical, it makes it clear that we can use standard CFG parsing algorithms. We conclude this section with a look at an alternative to statistical parsers

4.2.4 Transformation-based Parsing

Brill [6] has extended the transformation-based part-of-speech tagger described earlier (section 3) to construct a parser. He writes his parse tree structure in a style that owes its origins to the LISP programming language. In this notation our example parse (figure 5) is rendered as

$$((\text{ the cat }) (\text{ sat } (\text{ on } (\text{ the mat }))))$$

Here we have used the words of the sentence rather than the parts of speech.

The algorithm requires a naïve initialisation and a set of transformation rules to choose from.

initialisation The initialisation parse for our example sentence

$$(((((\text{ the cat }) \text{ sat }) \text{ on }) \text{ the }) \text{ mat })$$

fully specifies the method chosen.

set of rules The rules we choose affect the placement of the parentheses that determine the parse tree. We can use rules of the form

- add or delete a left or right parenthesis to the left or right of part-of-speech tag, t .
- add or delete a left or right parenthesis between of part-of-speech tags, t_i and t_j .
- add or delete a matching parenthesis to maintain the structural integrity of the parse tree.

The algorithm proceeds analogously to the transformational tagger earlier

1. parse naively
2. accumulate a list of rules that successively improve the parse
3. stop when additional rules bring minimal success

4.3 Data sets

To enable computers to parse human language using the methods described previously, we need databases (corpora) of language samples annotated to show their structural features, as a source of information and statistics to guide the development of language-processing algorithms. This in turn requires some set of parts of speech to be explicitly defined, so that researchers exchanging language data can be confident that they are using the annotations in the same way. Computational linguistics needs something like the Linnaean taxonomy created for botany in the 18th century, which for the first time enabled naturalists everywhere to exchange information about plants secure in the knowledge that when they used the same names they were talking about the same things.

A list of annotated texts for use in Statistical Language Processing is available at the University of Pennsylvania. This list shows how far we are from the aim of the previous paragraph with almost every corpus having a different annotation scheme. For parsing two of the largest are the Penn treebank and the SUSANNE corpus.

4.3.1 The Penn treebank

The Penn treebank [22] is the largest corpus of parsed sentences containing parse trees from the Brown Corpus which contains one million words from the Wall Street Journal. It is distributed by the Linguistic Data Consortium.

4.3.2 The SUSANNE corpus

The SUSANNE Corpus [28] contains annotations of a 130,000-word cross-section of written American English (it is based on a subset of the Brown Corpus). The annotation scheme used is an attempt to lay down a set of standards that sets out to be:

comprehensive covering all features of surface and logical English grammar that are definite enough to be susceptible of formal annotation, and including all phenomena that occur in practice in modern English

explicit if two researchers at separate sites are given the same sample of English and asked to annotate it according to the SUSANNE standards, their annotations should be identical

nonpartisan where aspects of grammar are the subject of theoretical controversy, the SUSANNE scheme aims to embody a neutral analysis which rival theoreticians can interpret in their own preferred terms

It is freely available from the Oxford Text Archive

5 Meaning

In the linguistics investigations into the meaning of language comes under the heading of semantics. The Oxford Shorter offers the following definition

semantic *adj.* relating to signification or meaning.

The subject is usually divided along word boundaries giving us:

lexical semantics concerning the meaning of individual words in context

structural or compositional semantics concerning how meanings combine to form larger meanings

We will discuss four approaches to semantics; collocations, Latent Semantic Analysis, logical forms.

5.1 Collocations

We are interested in discerning the meaning that a document communicates and want to present this information in compressed form. Such a summary reduces the amount of text to search and aids categorization.

It is tempting to think this information can be obtained that from considering the “interesting” words in a document, but there are cases where the individual words do not communicate the full weight of the meaning. An example of this is the phrase “international best practice” which on the surface appears to be just two adjectives modifying a noun in perfectly normal fashion. But, to any Australian reader of the last decade it has connotations, both business and political, well beyond its simple grammatical structure. Such a phrase needs a dictionary entry of its own.

We try and catch the essence in the idea of a collocation. A *collocation* is any turn of phrase or accepted usage where the whole has a meaning beyond the sum of its parts. Further examples to illustrate the concept include noun compounds like “disk drive”, which may be disk-shaped but does not drive anywhere, and “an English breakfast” which is highly suggestive of fried eggs and bacon.

The question of how to identify collocations is non-trivial even if we limit our search to those of length two.. The simplest approach to take would be to find the commonest two-word sequences in a text. Table 2 (from [21]) show the disappointing results of this process for a corpus consisting of stories in

Rank	Word 1	Word 2
1	of	the
2	in	the
3	to	the
4	on	the
5	for	the
6	and	the
7	that	the
8	at	the
9	to	be
10	in	a

Table 2: Most common bigram collocations in the 1990 *New York Times* corpus

the New York Times for 1990. The syntax of prepositional and noun phrases means that a preposition will often be followed by an article. Given the high frequency of both these parts of speech, it is unsurprising that these constructions top the list.

An obvious next step is to take into account the frequency of the individual words and deprecate common words in collocations. But the syntactic nature of the analysis gives us a clue how we could improve matters in a subtler manner. The most common part-of-speech patterns for bi-gram collocations are **Adjective Noun** and **Noun Noun** compounds. Table 3 shows the revised list after filtering for these two patterns. Almost all of them seem like good candidates for collocations. It is interesting to note that the top entry in this list, “New York”, was fifteenth in the previous list.

Apart from showing the importance of data in statistical natural-language processing for making our ideas falsifiable, this example shows how much syntactic information as simple as part-of-speech tags can add to the task of extracting meaning from textual data. Next we look at a way of taking into account the context in which words occur in deciphering documents.

5.2 Latent Semantic Analysis

From [17] Latent Semantic Analysis is a method for “extracting and inferring contextual-usage meaning of words”. The underlying idea is that words and phrases with a similar meaning will appear or not in a similar set of contexts.

It is a statistical technique that requires a large corpus of raw text, which

Rank	Word 1	Word 2	Part-of-speech pattern	
1	New	York	Adjective	Noun
2	United	States	Adjective	Noun
3	Los	Angeles	Noun	Noun
4	last	year	Adjective	Noun
5	Saudi	Arabia	Noun	Noun
6	last	week	Adjective	Noun
7	vice	president	Adjective	Noun
8	Persian	Gulf	Adjective	Noun
9	San	Francisco	Noun	Noun
10	President	Bush	Noun	Noun

Table 3: Most common bigram collocations in the 1990 *New York Times* corpus filtered by their part of speech

is parsed into words and separated into words and separated into meaningful passages; sentences and paragraphs. To illustrate the concept we have adapted the toy example from [17] in Table 4. The contexts are titles of some technical memos from two disparate literatures and we have italicized the words of interest, those that appear in more than two titles.

The first step is to construct a matrix indexed by the words of interest and the contexts. The entries are the frequency that the word appears in the context. This matrix for the example is shown in Figure 6. Before we continue, we take the logarithm of the incremented word frequencies and normalise each row by its entropy. If our initial matrix of frequencies is $F = (f_{ij})$ and we write $l_{ij} = \log(1 + f_{ij})$ then our new matrix F' has components

$$f'_{ij} = \frac{\log(l_{ij})}{\sum_i l_{ij} \log l_{ij}}$$

The effect of this transformation is to weight each word occurrence by its importance in the passage it occurs and inversely by the degree to which knowing a word tells you which context it appears in.

Next we apply singular value decomposition, a generalisation of eigenvalue analysis, to our $n \times m$ word frequency matrix, we can then write

$$F' = U \cdot D \cdot V^T$$

where U is an $m \times m$ matrix of left-eigenvalues, V is an $n \times n$ matrix of right-eigenvalues, and D is an $m \times n$ matrix which is zero except on its diagonal. These are the singular values.

- c1 *Human machine interface* for ABC computer applications.
- c2 A *survey* of *user* opinion of *computer system response time*
- c3 The *EPS user interface* management *system*
- c4 *System* and *human system* engineering testing of *EPS*
- c5 Relation of *user* perceived *response time* to error management

- g1 The generation of random, binary, ordered *trees*
- g2 The intersection *graph* of paths in *trees*
- g3 *Graph Minors* IV: Widths of *trees* and well-quasi-ordering
- g4 *Graph Minors*: A survey

Table 4: Data for toy example: Title of technical memos; c = human-computer interaction, g = graph theory

	c1	c2	c3	c4	c5	g1	g2	g3	g4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graphs	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Figure 6: The word by context matrix for our example.

The next step is to reconstruct using fewer dimensions by setting all but the two largest singular values to zero in D giving \bar{D} and calculating $R = U \cdot \bar{D} \cdot V^T$. For our example R is shown in figure 7. The number of dimensions to reduce by is a parameter of the LSA and must be chosen by some mechanism. In this case the choice of two is suggested by a natural grouping of the contexts.

It is interesting to note the changes to the frequency of the words “survey” and “trees” in the title of the fourth graph theory article between figures 6 and 7. The word “survey” appears in context g4 but its appearance in the c2 context means it cannot be associated with a single group. This is reflected in the reduction of its notional frequency to 0.42 after the dimensional reduction step. Conversely “tree” which is closely aligned with the graph theory group but doesn’t appear in the g4 context has its frequency raised to 0.66 from 0.

LSA, as it is currently practiced, makes no use of word order, syntax, logic, morphology in its analyses of meaning. This does not stop its more enthusiastic advocates claiming that it models the computational processes and representation underlying a substantial portions of the acquisition and utilisation of knowledge by human beings. In support of their claims they offer, as evidence, tasks where LSA has performed equivalently to human subjects. Some examples include

1. synonym judgments
2. restricted subject matter multiple-choice tests
3. subjective ranking of text properties (*e.g.* marking essays)
4. predictor of query-document similarity judgments

This last area is closest to our current interest is the last. In this context the technique is known as Latent Semantic Indexing (LSI) [11]. We form the term versus document matrix as before. Usually the document is represented by its title, abstract and/or a keyword list. A query is then represented as a pseudo-document, a weighted average of the word vectors it contains. These word vectors are calculated as weighted averages of the document vectors that the word are contained in. Document retrieval can be achieved by returning the documents with the highest frequency in the query vector.

5.3 Logical Forms

The approaches that we have just detailed are rather *ad hoc*. A little rigour can be added by the use of the logical form for sentences [23]. Logical forms

arose in the context of attempts to say more about the intuitive distinction between inferences like

Socrates is a man.
All men are mortal.
Therefore Socrates is mortal

and riskier ones such as

Socrates is a man.
Socrates is mortal.
Therefore all men are mortal.

Both are syntactically correct but the the second is clearly unsound.

From the the time of Aristotle logic was practiced using natural language, exactly as in the previous paragraph. It wasn't until Frege [12] distinguished logical from grammatical form in 1879. The modern language of logic, First Order Predicate Calculus (FOPC), is, essentially, his system of logic with notational changes. An illustration of his notation and a demonstration of it unworkability can be found on the cover of Russell and Norvig [26].

Frege's main contribution was to assert that propositions, for example

Socrates is a man,

have a predicate-argument structure, *i.e.*

$male(\text{Socrates})$

rather than the subject-predicate form that is suggested by the English representation. It is ironic, that having freed logic from the constraints of natural language, we would use this highly-developed machinery for manipulating truth values to help understand the meaning conveyed by natural language utterances.

We would like to show how the meaning of a complex expression depends on

- the meaning of its constituents, and
- the arrangement of these constituents

The meaning of a sentence is determined by its logical form and the logical form of its constituent propositions, we say a sentence, S, means the logical form, p, if our set of rules assigns the truth value TRUE if p and false otherwise.

The analysis just completed assumes that we can generate a logical form for a sentence. We show this is possible by following the following steps.

1. We exhibit a translation from our rule-based grammars to FOPC.
2. Given this infrastructure, we can augment the grammar with semantic relations, which run in parallel to the syntactic relations.
3. We show that it is not possible to construct a pure logical form from a grammar and introduce an form intermediate between the syntactic and semantic, called quasi-logical form.

5.3.1 Definite-clause grammars

It is possible to write the production rules used in the grammars we displayed in Section 4.1 using the predicate calculus:

Rule	Logical sentence
$S \rightarrow \text{NP VP}$	$NP(s_1) \wedge VP(s_2) \Rightarrow S(\text{Append}(s_1, s_2))$

We can take the logical sentence to say that if there is a string, s_1 that is a noun phrase and a string s_2 that is a verb phrase then the string formed by appending them is a sentence, which is equivalent to the rule. These strings are the pieces of the sentence that are being identified.

Arbitrary logical sentences are too costly to calculate with so we constrain ourselves to using definite clauses. When written as an implication, definite clauses have a consequent consisting of a single atom, C , and a conjunction of zero or more atoms in the antecedent, thus

$$A_1 \wedge \dots \wedge A_n \Rightarrow C$$

A grammar constructed in this fashion is called a *definite-clause grammar* (DCG). We can set up a translation table so we can continue to use our old rule notation

Logical sentence	Rule
$Y(s_1) \wedge Z(s_2) \wedge \dots \Rightarrow X(\text{Append}(s_1, s_2, \dots))$	$X \rightarrow Y Z \dots$
$X([\text{“word”}])$	$X \rightarrow \text{word}$
$Y(s) \vee Z(s) \vee \dots \Rightarrow X(s)$	$X \rightarrow Y \mid Z \mid \dots$

5.3.2 Semantics as DCG Augmentations

Non-terminal symbols in a grammar can be augmented with extra arguments that can be used to carry semantic information.

To express a noun phrase with semantics “*sem*” we write, in DCG form, $NP(\text{sem}, s)$ or just $NP(\text{sem})$ in a rule. We allow variable on the right-hand side of rule, for example, a rule that identifies any repeated words

rule : $\text{Double} \rightarrow ww$
logic : $s_1 = [w] \wedge s_2 = [w] \Rightarrow \text{Double}(\text{Append}(s_1, s_2))$

Arbitrary logical text we enclose in braces $\{\}$.

As an example we display in Figure 8 a DCG specifying the syntax and semantics for a tiny fragment of English. For the moment we will ignore Rules 3–5 and 10, and show how to derive the logical form $\text{hears}(\text{tom}, \text{sam})$ from the sentence “Tom hears Sam.” The semantics of the individual parts of the sentence are easily established

Tom has semantics tom using Rule 6

Sam has semantics sam using Rule 7

hears has semantics $\lambda y \lambda x \text{Hears}(x, y)$ using Rule 8

Rule 2 describes the syntax of a verb phrase as a combination a verb and a noun phrase, as before. Additionally, it now states that the semantics of the verb phrase “hears Sam” is obtained by applying the semantics of the verb, “hears”, to the semantics of the noun phrase “Sam” (*i.e.* $\lambda y \lambda x \text{Hears}(x, y)(\text{sam})$) or, equivalently, $\lambda x \text{Hears}(x, \text{sam})$). We repeat this process for Rule 1 and pass the semantics of the noun phrase “Tom” as the argument of the semantics of the verb phrase “hears Sam” yielding $\text{hears}(\text{tom}, \text{sam})$

5.3.3 Quasi-Logical Form

The last example was fairly straightforward, but consider the sentence

Everyone hears Sam.

for which,

$$\forall a \text{Hears}(a, \text{sam}),$$

seems a reasonable semantic interpretation. However, when we try and build this interpretation from the constituents of the sentence,

$$\begin{array}{ll} \text{Everyone} & \text{NP}(\forall a \dots) \\ \text{hears Sam} & \text{VP}(\text{Hears}(\dots, \text{sam})), \end{array}$$

two problems emerge. Firstly, the semantics of the sentence would appear to consist of the semantics of the NP with the the semantics of the VP replacing the ellipsis, which is inconsistent with Rule 1. Second, the quantification variable a is required as an argument of the relation Hears . In order to specify the semantics of this sentence we appear to need a couple of potentially very confusing intertwined functional compositions.

	c1	c2	c3	c4	c5	g1	g2	g3	g4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graphs	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

Figure 7: The reconstructed word by context matrix for our example *after* dimensional reduction.

1. $S(\text{rel}(\text{obj})) \rightarrow \text{NP}(\text{obj}) \text{VP}(\text{rel})$
2. $\text{VP}(\text{rel}(\text{obj})) \rightarrow$
 $\text{Verb}(\text{rel}) \text{NP}(\text{obj})$
3. $\text{NP}(\text{ref}(\text{he})) \rightarrow \mathbf{he} \mid \mathbf{him}$
4. $\text{NP}([\exists a a]) \rightarrow \mathbf{someone}$
5. $\text{NP}([\forall a a]) \rightarrow \mathbf{everyone}$
6. $\text{NP}(\text{tom}) \rightarrow \mathbf{Tom}$
7. $\text{NP}(\text{sam}) \rightarrow \mathbf{Sam}$
8. $\text{Verb}(\lambda x \lambda y \text{Hears}(x, y)) \rightarrow$
 \mathbf{hears}
9. $\text{Verb}(\lambda x \text{Sneezes}(x)) \rightarrow \mathbf{sneezes}$
10. $\text{Verb}(\text{ellipsis}) \rightarrow \mathbf{did}$

Figure 8: A grammar specifying both syntax and semantics.

The problem arises because of the same tension between syntactic and semantic structures that hampered logic before Frege. To solve this problem we augment our logical form with a new construction called a *quantified term* for expressing the semantics of constructions like “everyone” and “everything”. The semantic interpretation for the former is given in Rule 5 as the quantifier surrounded by square brackets and the semantics of our example can be written, now in *quasi-logical form* intermediate between syntax and semantics, as $Hears([\forall a], sam)$.

Now we have an additional step of converting the quasi-logical form into real first-order logic, but to do this successfully requires more information than is available in the sentence alone. We will defer consideration of the this issue until our discussion of context in section 6.1.

6 Context

In logic conjunction is commutative:

$$P \wedge Q \wedge R \equiv R \wedge P \wedge Q.$$

but this is not true for natural language as the following example shows

Holiday A	Holiday B
I went to Siberia.	I went to Siberia.
I looked for polar bears	I lay on the beach.
Then I went to Hawaii.	Then I went to Hawaii.
I lay on the beach.	I looked for polar bears.

Even though both holidays are described with the same sentences, it abundantly clear which was the more successful. The variant orderings of the sentences have set up different contexts in which the subsequent sentences are interpreted.

Another interpretation of the discourse concerning Holiday B above is that it describes the same events as Holiday B, but the text has been incorrectly constructed. That this trip seems farcical highlights the fact that we expect text to be *coherent*, without any non-sequiturs or gaps.

The study of such phenomena is part of the field of *pragmatics*. All of the work described so far in this report has used only isolated sentences. In this section we begin to broaden our horizons and look at collocated, related groups of sentences. Such a group is termed a *discourse* and the study of the interrelationships among them is called *discourse processing*.

We will cover three areas within this field. We will firstly show that the conversion from the quasi-logical form of section 5.3 to real first-order logic

is ambiguous and therefore depends on the context. Next, we will describe an approach to resolving the unspecified parts of semantic interpretations. Finally, we describe Rhetorical Structure Theory (RST) [20] as one method of analysing the coherence structure of a discourse.

6.1 Converting from quasi-logical form to logical form

To convert from quasi-logical form (QLF) into logical form we must replace the quantified terms by real terms. This can be done by a simple rule for each quantifier. If $U = [\forall x P(x)]$ is a universally quantified term in the QLF, $Q(U)$ then we can rewrite using the rule

$$Q(U) \rightarrow \forall x P(x) \Rightarrow Q(x).$$

For an existentially quantified term $E = [\exists x P(x)]$ the equivalent rule is

$$Q(E) \rightarrow \exists x P(x) \wedge Q(x).$$

We are essentially pulling the quantifier out the front and replacing the quantified term with the quantification variable.

But, there is a sting in the tail of this deceptively simple solution. Consider the sentence

Everyone hears someone.

which has a QLF

$$Hears([\forall a a], [\exists b b]).$$

The rules specify how to convert these quantified terms but not which order to apply them. This gives rise to two resolved logical forms for our sentence

$$\forall a (\exists b Hears(a, b))$$

and

$$\exists b (\forall a Hears(a, b)).$$

with the difference being whether everyone heard the one person or a person of their own. This ambiguity can only be resolved by looking at the context in which the sentence occur and discovering who the “someone” being referred to is.

Sentences	QLFs
Tom sneezes.	$sneeze(tom)$
Sam hears him.	$hears(sam, ref(he))$
Everyone did.	$ellipsis([\forall a a])$

Table 5: Three contextually dependent sentences and their QLFs.

6.2 Reference Resolution

The context independent meaning of a sentence is contained in one or more quasi-logical sentences. These sentences also contain elements that depend on the context; pronouns, ellided verbs *etc.*

Contextual resolution is a process of adding information to an underspecified meaning representation until it can be useful. A system that uses this process extensively is the Core Language Engine (CLE) [3], where the contextual processing is a larger part of the system than either the syntactic or semantic processing.

We will describe the approach to reference resolution proposed by Pulman [25] and show its application to identifying the referents of pronouns and reinstatement of verbs that have been elided. This approach improves on that used by the CLE, detailed in [4], by removing unnecessary indices and metavariables and a simplified relationship between the QLF and its resolved form.

We represent the transition between the context dependent QLF and a resolved logical form (RLF) with at least one less dependency as a *conditional equivalence* of the form

$$\begin{array}{c}
 \text{QLF} \iff \text{RLF} \\
 \text{if} \\
 \textit{condition}_1 \\
 \dots \\
 \textit{condition}_n
 \end{array}$$

An equivalence like this can be paraphrased as: “In this context this QLF can be interpreted as this RLF if and only if the goals, *condition*₁ to *conditions*₂, can be satisfied.” These goals are treated as logic programs and higher-order matching is used to determine whether a particular equivalence applies.

To illustrate our approach with an example we return to the semantic grammar of figure 8 and consider the three sentences and their QLFs in Table 5. The second and third sentences contain contextual dependencies; a

pronoun, $ref(he)$, and an elided verb, $ellipsis$.

A simple conditional equivalence for resolving pronoun reference is:

$$\begin{aligned}
 Verb(ref(Pro)) &\iff Verb(Someone) \\
 \text{if} & \\
 &context(C) \\
 &OtherVerb(Someone) = C \\
 &Pro(Someone)
 \end{aligned}$$

We have a match for the QLF of the second sentence; $Verb = \lambda x hears(sam, x)$ and $Pro = he$. Here we see the higher-order matching at work where we have allowed a variable to take a function as its value. The first condition succeeds and identifies the context, $C = sneeze(tom)$. The second extracts the subject with the successful matches; $OtherVerb = sneeze$ and $Someone = tom$. Finally $he(tom)$ succeeds as Tom is indeed a he and the corresponding RLF is $(\lambda x hears(sam, x))(tom)$ which reduces to $hears(sam, tom)$.

To resolve the ellipsis in the third sentence we need another conditional equivalence:

$$\begin{aligned}
 ellipsis(X) &\iff Verb(X) \\
 \text{if} & \\
 &context(C) \\
 &Verb(Subject) = C \\
 ¶llel(X, Someone).
 \end{aligned}$$

In our example matching gives $X = [\forall a a]$ and $C = hears(sam, tom)$ after the analysis of the previous paragraph. The second condition yields $Verb = \lambda x hears(x, sam)$ and $Subject = sam$. The predicate $parallel$ succeeds if its two arguments can be used in similar places. So if we assume that everyone can do what Sam does then the logical form of the sentence becomes

$$\forall a hears(a, tom)$$

after resolution of the quantified term.

It is interesting to note the implication in the conditional equivalences runs both ways so from the sequence of logical forms

$$\begin{aligned}
 &sneeze(tom) \\
 &heard(sam, tom)
 \end{aligned}$$

we could generate the sentences ‘‘Tom sneezes. Sam hears him.’’ by running the process in reverse. The process is, in principle, completely reversible.

1. Lactose and Lactase
2. Lactose is milk sugar
3. the enzyme lactase breaks it down
4. For want of lactase most adults cannot digest milk
5. I populations that drink milk th adults have more lactase perhaps through natural selection.

Figure 9: Title and Abstract from an article from the October 1972 edition of Scientific American

6.3 Rhetorical Structure Theory

RST was originally developed for computer based text generation; it describes texts rather than modelling their creation or interpretation. It posits an evident rôle for each part of a coherent text, a plausible reason for its presence. It is the basis for the discourse processing aspects of Spärck Jones' work [30] on automatic summaries.

There are two basic building blocks in this description, the first being the *span*. A span may be a sentence or just a phrase. Figure 9 shows an excerpt from a Scientific American article broken into spans. The second is a set of *coherence relations* between the spans. A span that makes a claim followed by a span that provides evidence to support the claim gives rise to an *evidence relation*. As the claim is the more important it is termed the *nucleus* of the relation and the evidence span is the *satellite*. Table 6 shows some of the other possible relations. A relation whose spans are of equal importance is called *multinuclear* and we give the example of the neutral contrast relation. Using the spans and relations defined so far we can perform an RST *analysis* that arranges the spans into a binary tree structure whose leaves are the spans and the nodes are the relations between them.

Figure 10 shows the RST analysis for the Scientific American article in traditional format. The analysis is constructed by an *observer* who is distinguished from both the author and the reader of the text. The use of a relations asserts beliefs of the observer about the author's own beliefs and intentions towards the reader. The use of the Background relation linking the satellite spans 2 and 3 to the nuclear spans 4 and 5 has implicit in it the following beliefs of the observer

Relation	Nucleus	Satellite
Background	text whose understanding is being facilitated	text for facilitating understanding
Elaboration	basic information	additional information
Preparation	text to be presented	text which prepares reader to expect and interpret text to be presented
Multinuclear	Span	Other Span
Contrast	one alternate	the other alternate

Table 6: Examples of RST coherence relations

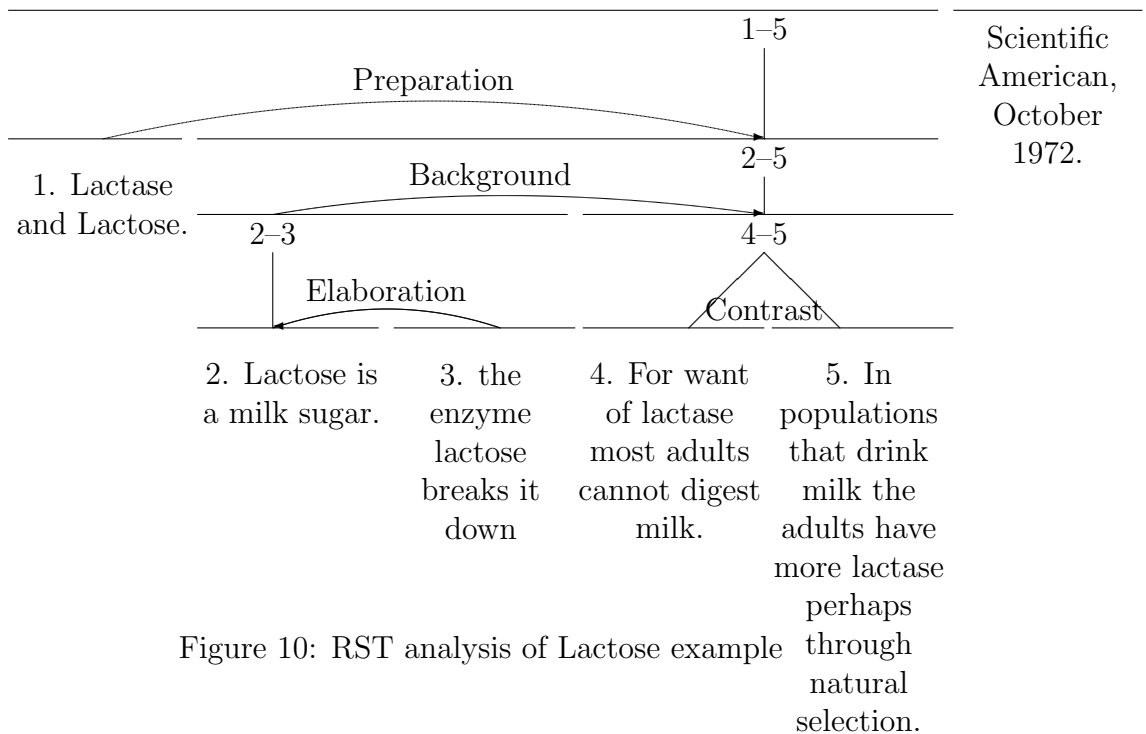


Figure 10: RST analysis of Lactose example

- The reader may not understand the nucleus to a degree satisfactory to the author.
- The author believes reader will understand the satellite.
- The author believes that understanding the satellite will facilitate understanding the nucleus.
- The reader's understanding of the nucleus is facilitated.

To summarise, RST provides a motivated account of why each element of a coherent text was included by its author. The account is independent of lexical and grammatical forms and augmented by a notional observer that allows the subjectivity or otherwise of the analysis to be considered.

7 Research Directions

So what is the core problem? It is a perfectly familiar one, namely how language and the world are related. [31]

This quote from Karen Spärck Jones' address to the Association for Computational Linguistics identifies the issue that is central to any understanding of language, but that is most often finessed using corpus-based linguistic data or domain-specific semantic patterns. We will discuss how the Business Intelligence Group's focus on information integration provides a mechanism for facing this problem squarely. We lay out a conceptual framework that brings the goal of integrating text corpora and relational databases and detail some specific research projects needed to help realise this structure.

7.1 Information Integration

The key research problem in Business Intelligence is knowledge integration and analysis (KIA). There exists a number of reasonably well-developed technologies for interpreting various data modes in terms of relatively low-level constructs¹. The next step is to develop models that are sufficiently rich, user-friendly, fast and practical to allow inferences to be made across modes.

Language is used to convey messages about entities and concepts that are usually not part of the language representation itself. The KIA approach would allow us to carry non-linguistic representations of the concepts and entities alongside the documents that refer to them. Reasoning could be

¹For example, tabulated values, text, imagery, *etc.*

performed not only about the text but also on the actual thing itself. This ability would be unique in NLP systems and would be a first step in using explicitly the relationship between language and its real-world referents.

7.2 Active Summaries

Traditional relational database systems are not specifically structured for querying and reporting. So BI systems often recreate the databases as multidimensional indices by tracing paths through the relational structure. To avoid confusion with the wider definition of BI (section 1) we will refer to this as on-line analytical processing (OLAP). These data warehouses are often known as OLAP cubes or simply cubes.

Developing OLAP applications packages is conceptually straightforward. You require access to the Enterprise Resource Management (ERM) and Human Resources (HR) relational databases via an structured query language (SQL) server, an OLAP database generator, a middleware system (e.g. CORBA or COM), a data mining/statistics tool kit, a drag and drop interface and an interactive web application server with strong spreadsheet and graphics capabilities. With such tools one can create systems such as

- slice and dice, drill-down financial exploration tools
- customer profiling systems
- direct mail optimising systems
- balanced score cards
- portals
- fast drag and drop data-mining tools
- filtered wire services (e.g. Reuters, Dow-Jones)

The idea of an active summary is to replicate this infrastructure for a collection of text documents. This would involve identifying the semantic relations within and between documents and generating multidimensional views to facilitate the generation of reports and answering of queries in a text database. The use of the familiar OLAP paradigm would ease adoption of active summary products in areas already familiar with OLAP technologies.

7.3 Research Projects

The strength of the BI Group lies in its ability to develop effective probabilistic algorithms for complicated statistical inference problems. Although promising, work on Bayesian networks and rule-based processing has not produced effective methods that scale and handle uncertainty.

The techniques we have surveyed in this report can be identified with the type of information they handle:

symbolic techniques use only the representational form of the text (characters and words)

syntactic techniques use grammatical information, parts of speech, sentence parses *etc.*

semantic techniques use some representation of the meaning of the text being mined

discursive techniques use contextual information in their operation.

The use of probabilistic models is quite prevalent in the first two of these areas but almost non-existent in the study of semantics and discourse allowing us to make a significant contribution by developing a statistical approach to these areas. The active summaries idea hinted at above allows us to explore intradocument structure at paragraph level and above as well as interdocument coreferences, two important properties that have received scant attention so far in the literature.

References

- [1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: principles, techniques and tools*. Addison-Wesley, 1986.
- [2] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, 1972.
- [3] H. Alshawi, editor. *The Core Language Engine*. MIT Press, Cambridge, MA, 1992.
- [4] H. Alshawi and Crouch. Monotonic semantic interpretation. In *Proceedings of the 30th Annual Meeting, Association for Computational Linguistics*, pages 32–39, 1992.
- [5] E. Brill. A simple rule-based part-of-speech tagger. In *Proc. Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.
- [6] E. Brill. Automatic grammar induction and parsing free text: a transformation-based approach. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 259–265, 1993.
- [7] E. Charniak. *Statistical Language Learning*. MIT Press, 1993.
- [8] E. Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18(4):33–43, 1997.
- [9] N. Chomsky. *Syntactic Structures*. Mouton, The Hague and Paris, 1957.
- [10] C. L. A. Clarke and G. V. Cormack. On the use of regular expressions for searching text. Technical Report CS-95-07, Department of Computer Science, University of Waterloo, 1995.
- [11] S. T. Dumais. Latent semantic indexing (LSI) and TREC2. In D. Harman, editor, *The Second Text Retrieval Conference (TREC2)*, pages 105–116, 1994. National Institute of Standards and Technology Special Publication 500—215.
- [12] G. Frege. Begriffsschripte. In J. van Heijenoord, editor, *From Frege to Gödel : A sourcebook of mathematical logic*. Harvard, 1967.
- [13] A. K. Joshi and Y. Schabes. Tree-adjointing grammars. In G. Rosenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–123. Springer Verlag, 1996.

- [14] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2000.
- [15] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.
- [16] R. Krovetz. Viewing morphology as an inference proces. In *Proceedings of ACM-SIGIR93*, pages 191–203, 1993.
- [17] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, (25):259–284, 1998.
- [18] W. Little, H. W. Fowler, and J. Coulson. *The Shorter Oxford English Dictionary on Historical Principles*. Oxford University Press, third edition, 1964.
- [19] D. M. Magermann. Statistical decision-tree models for parsing. In *ACL-95*, pages 276–283. Association for Computational Linguistics, 1995.
- [20] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Towards a functional theory of text organisation. *Text*, 8(3):243–281, 1988.
- [21] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- [22] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building and large annotated corpus of English: the Penn treebank. *Computational Linguistics*, (19):313–330, 1993.
- [23] P. M. Pietroski. Logical form. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Laboratory, CSLI Publications, spring edition, 2000. URL: <http://plato.stanford.edu/archives/spr2000/entries/logical-form/>.
- [24] Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [25] S. G. Pulman. A computational theory of context dependence. Technical Report CRC048, SRI International Cambridge Computer Science Research Centre, 1994.
- [26] S. J. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice Hall, 1995.

- [27] I. A. Sag and T. Wasow. *Syntactic Theory: A Formal Introduction*. Stanford: CSLI publications, 1999.
- [28] G. Sampson. *English for the computer*. Clarendon Press, 1995.
- [29] S. M. Schieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, (8):333–343, 1985.
- [30] K. Spärck Jones. What might be in a summary? In Knorz, Krause, and Womser-Hacker, editors, *Information Retrieval 93: Von der Modellierung zur Anwendung*, pages 9–26. Universitätsverlag Konstanz, 1993.
- [31] K. Spärck Jones. Natural language processing: She needs something old and something new (maybe something borrowed and something blue, too.). In *Presidential Address*. Association for Computational Linguistics, 1994.
- [32] K. Thompson. Regular expression search algorithms. *Communications of the Association for Computing Machinery*, 11(6):419–422, 1968.
- [33] K. Vijay-Shanker and D. Weir. The equivalence of four extensions to context-free grammars. *Mathematical Systems Theory*, (27):511–546, 1995.
- [34] XTAG-Group. A lexicalized tree-adjoining grammar of english. Technical Report 95-03, Institute for Research in Cognitive Science, University of Pennsylvania, 1995.