

IMPROVING TOPIC TRACKING USING DOMAIN CHAINING

Li Yang

Thesis Prepared for the Degree of

MASTER OF ARTS

UNIVERSITY OF NORTH TEXAS

August 2003

APPROVED:

Timothy Montler, Major Professor  
John Robert Ross, Committee Member  
Rada Mihalcea, Committee Member  
James Tanner, Chair of the Department of English  
C. Neal Tate, Dean of the Robert B. Toulouse  
School of Graduate Studies

Yang, Li, *Improving Topic Tracking with Domain Chaining*. Master of Arts (Linguistics), August 2003, 94 pp., 41 titles.

Topic Detection and Tracking (TDT) research has produced some successful statistical tracking systems. While lexical chaining, a non-statistical approach, has also been applied to the task of tracking by Carthy and Stokes for the 2001 TDT evaluation, an efficient tracking system based on this technology has yet to be developed. In thesis we investigate two new techniques which can improve Carthy's original design.

First, at the core of our system is a semantic domain chainer. This chainer relies not only on the WordNet database for semantic relationships but also on Magnini's semantic domain database, which is an extension of WordNet. The domain-chaining algorithm is a linear algorithm. Second, to handle proper nouns, we gather all of the ones that occur in a news story together in a chain reserved for proper nouns.

In this thesis we also discuss the linguistic limitations of lexical chainers to represent textual meaning.

## ACKNOWLEDGMENTS

I am grateful to Dr. Mihalcea, Dr. Ross, and Dr. Montler for being great advisors, professors and mentors.

## CONTENTS

1	INTRODUCTION	1
2	TOPIC TRACKING	5
2.1	Topic Detection and Tracking . . . . .	5
2.2	TDT Terminology and Corpora . . . . .	6
2.3	TDT Evaluation Methodology . . . . .	7
2.4	Topic Tracking . . . . .	9
2.4.1	Tracking using Lexical Chaining . . . . .	10
2.5	TREC 2002 Filtering Track . . . . .	11
2.5.1	Definitions and Evaluation . . . . .	11
2.6	TDT and TREC Tracking . . . . .	14
3	ISSUES WITH EARLIER LEXICAL-CHAINING ALGORITHMS	16
3.1	Meaning of Text and Lexical Cohesion . . . . .	16
3.2	Definition of Lexical Chain . . . . .	18
3.3	A Generic Chaining Algorithm . . . . .	20
3.4	WordNet and the Chain Computing . . . . .	21
3.4.1	Computing Relatedness . . . . .	21
3.4.2	Potential Problems . . . . .	22
3.5	Issues with Earlier Chainers . . . . .	23
3.5.1	Stairmand's Chainer for IR . . . . .	23
3.5.2	Greedy Chainers . . . . .	24

3.6	St-Onge's Chainer . . . . .	25
3.6.1	The Algorithm . . . . .	25
3.6.2	A Sample Text and its Intuitively Natural Chains . . . . .	26
3.6.3	Over and Under Chaining . . . . .	28
3.6.4	Suggestion for Improvement . . . . .	29
3.7	Summary of the Issues with Early Chaining Algorithms . . . . .	29
4	IMPROVING THE CHAINING PROCESS . . . . .	31
4.1	Barzilay's Dynamic Chaining Algorithm . . . . .	31
4.1.1	Comments on St-Onge's Algorithm . . . . .	32
4.1.2	Barzilay's Algorithm in Brief . . . . .	33
4.1.3	Lessons Learned . . . . .	34
4.2	Silber and McCoy's Linear Algorithm . . . . .	34
4.2.1	The Linear Algorithm . . . . .	34
4.2.2	Modifying the WordNet Noun Database . . . . .	35
4.2.3	Advantages of the Linear Algorithm . . . . .	35
4.3	Carthy's Chainer . . . . .	36
4.4	Modifying the WordNet Database . . . . .	36
5	CHAINING WITH SEMANTIC DOMAINS . . . . .	40
5.1	Semantic Domains and Lexical Chains . . . . .	40
5.2	Domain Lexical Chainer . . . . .	41
5.3	Handling Proper Nouns . . . . .	41
5.4	Building Chains for a Sample Story . . . . .	43

6	APPLYING CHAINING TO TRACKING	47
6.1	Tracking with Carthy's System . . . . .	47
6.1.1	Handling Proper Nouns . . . . .	49
6.2	Implementing Carthy's Tracking System . . . . .	49
6.2.1	Data Structures . . . . .	50
6.2.2	Functions . . . . .	50
6.3	Tracking with Domain Chaining . . . . .	52
6.3.1	Computing Similarity . . . . .	53
6.4	Implementing Domain-Tracking System . . . . .	54
6.4.1	Data Structures . . . . .	54
6.4.2	Functions . . . . .	54
6.5	Hypotheses . . . . .	56
7	EXPERIMENT AND EVALUATION RESULTS	59
7.1	Comparing Chaining Results . . . . .	59
7.2	Comparing Tracking Systems . . . . .	61
7.2.1	$T11SU$ and $T11F$ Scores . . . . .	62
7.2.2	Precision-Recall and DET Plots . . . . .	63
7.2.3	Average Scores . . . . .	63
7.2.4	Comparing Average T11SU Scores . . . . .	64
7.2.5	Comparing Average T11F Scores . . . . .	65
7.2.6	Precision and Recall Scores . . . . .	66
7.2.7	Average $P_{miss} - P_{fa}$ Scores . . . . .	67

8	CAN LEXICAL CHAINS REPRESENT TEXTUAL MEANING	69
8.1	Cohesion Helps to Create Text . . . . .	70
8.1.1	Cohesive Resources . . . . .	70
8.1.2	Cohesion as a Necessary Condition for Meaning . . . . .	71
8.1.3	Lexical Chains and Textual Meaning . . . . .	72
8.2	Static and Dynamic Ties . . . . .	73
8.3	Discourse Semantic Structure . . . . .	75
8.3.1	Systems of Discourse Structure . . . . .	75
8.3.2	Textual Meaning Model . . . . .	77
8.3.3	Positioning Lexical Chains . . . . .	78
9	CONTRIBUTIONS, LIMITATIONS AND FUTURE WORK	79
9.1	Contributions . . . . .	79
9.1.1	The Chaining Algorithm . . . . .	79
9.1.2	Tracking . . . . .	79
9.2	Limitations . . . . .	79
9.2.1	Semantic Domains . . . . .	80
9.2.2	Tracking . . . . .	80
9.3	Future Work . . . . .	81
9.3.1	Semantic Domains . . . . .	81
9.3.2	Combining Statistical and Lexical Chaining Approaches . . . . .	81
9.3.3	Evaluations . . . . .	81
A	CHAINS FROM THE SAMPLE TEXT	82

B MORE T11SU AND T11F FIGURES	85
C MORE PRECISION-RECALL AND DET PLOTS	88
D SYSTEM AVERAGE SCORES	90
BIBLIOGRAPHY	91



## CHAPTER 1

### INTRODUCTION

In the last five years, much research has been done to investigate statistical approaches to the task of topic detection and tracking (TDT); several successful statistical systems have been produced. While such systems are the major systems for topic tracking, Joe Carthy, in his 2001 TDT evaluation, was the first to propose a tracking system based on lexical chaining, a shallow natural language processing technique. Lexical chaining techniques have been around for about a decade now, since Morris and Hirst's seminal research on computing lexical chains in 1991.

According to Morris and Hirst [31], a lexical chain is a sequence of semantically related words in a text, which creates a context and contributes to the continuity of meaning. A lexical chain may span short distances (adjacent words or sentences) or long ones (entire texts), and is independent of the grammatical structure of the text. A chaining process is one which extracts a list of words that captures a portion of the cohesive structure of the text. The part of the system that performs the chaining process is called a chainer.

Since the mid-90's lexical chaining has been applied to fields such as information retrieval, text summarization, hypertext linking, malapropism detection and multimedia indexing. St-Onge wrote a chainer for malapropism detection and correction in 1995 [37]. Kazman and Al-Halimi made a chainer for multimedia indexing [22]. Green used a chainer to build hypertext links [15]. Among the chainers for text summarization are Stairmand's [38], Barziley's [5], and Silber and McCoy's [36].

Lexical chainers in all these applications rely on WordNet semantic relationships between lexical items. However, all these chainers, Joe Carthy’s included, face the following two difficulties.

First, these chainers rely on the semantic relationships provided by WordNet in their chaining processes. However, since WordNet does not provide for semantic relationships between all lexical items, such as items related by non-systematic semantic relationships, lexical chainers depending only on the WordNet database cannot effectively extract chains from a story.

Secondly, at present, the TDT program requires that tracking systems be designed to handle event-based news stories. Each news story is associated with some topic. A TDT corpus defines a topic as an event that has a five-tuple of components,  $\{who, when, where, what, why\}$ . But proper nouns are also valuable, because they make up the *who*, *when*, and *where* components of the five-tuple. However, previous lexical chainers did not have effective techniques for handling proper nouns, partly because WordNet prior to version 1.7 did not include proper nouns in its database. Lexical chainers without an appropriate methodology for handling proper nouns cannot effectively extract chains from a story.

In this thesis, we will present a tracking system along the lines of Joe Carthy’s tracking system. However, the proposed system is different in two important ways. First, we propose a semantic-domain-based chaining algorithm as an improvement over previous lexical chainers, specifically Joe Carthy’s chainer. Secondly, we propose a technique for handling proper nouns.

At the core of our tracking system is a semantic domain chainer. This chainer relies

not only on the WordNet database for semantic relationships but also on Magnini’s semantic domain database, which is an extension of WordNet. The domain chainer stems from Mihalcea’s notion of the “coarse granularity” of WordNet senses, and will only assemble lexical items in the same domain into a lexical chain. This restriction improves the lexical chaining process and tracking results. Importantly, the domain-chaining algorithm is a linear algorithm.

To handle proper nouns, we assemble all the proper nouns in a news story into a chain reserved only for proper nouns. The proper noun collecting process does not use WordNet or Magnini’s database. Instead, it uses Brill’s tagger to find all the proper nouns in a story. This methodology improves not only the tracking system but also its running time.

To compare our tracking system with Carthy’s and to test the proper-noun-handling technique, we have implemented both Carthy’s and our tracking systems and have run both systems on the training and testing files for the 100 topics used for the 2002 Text REtrieval Conference (TREC) evaluation. When evaluating the systems, we used both TDT and TREC evaluation measures. By the utility scores of the TREC measures, the domain-tracking system outperformed Carthy’s system, whether with or without proper-noun handling techniques. Since we did not use the fourth TDT corpus (TDT-4), the corpus for the 2002 TDT evaluation, although we provide the results by TDT measures, we will not comment on these results.

Each lexical chainer we report on in this thesis makes an effort to improve some previous chaining algorithm in terms of using more efficient tools or chaining algorithms. There has been significant progress in later lexical chainers. However, all chainers base their linguistics background on the assumption that lexical cohesion per se represents textual meaning.

Therefore, by extracting lexical chains themselves, a chainer extracts the meaning of a document. Unfortunately, textual meaning is based on more complicated techniques, and thus more effort than extracting lexical chains is required to represent textual meaning. We explain this in detail in this thesis.

The thesis is organized as follows: we first describe the terminology, corpora, and evaluation methodologies of the TDT tracking task, compare them with those of the 2002 TREC filtering track task, and explain why we have used both TDT and TREC evaluations in this work. Chapter 3 describes the linguistic background for and computability of lexical chains using WordNet, and examines two types of problems with previous chainers. Chapter 4 describes the efforts that have been made in later lexical chainers to improve the chaining algorithms in earlier chainers, and also the work that has been done to modify the WordNet database for better chaining results. In chapter 5, we present our domain-based chainer, using Magnini’s domains as an extension to WordNet. In chapter 6 we present an implementation of both Carthy’s and our own domain-chaining-based tracking systems. Chapter 7 presents the results of four sets of experiments. Chapter 8 explains why lexical chains cannot completely represent document meaning. We discuss future work in chapter 9.

## CHAPTER 2

### TOPIC TRACKING

In this thesis, we propose a domain-chaining tracking system, as an improvement on the topic-tracking system using lexical chains originally put forward by Joe Carthy [11]. Since our work in this thesis is one of the subtasks of the Topic Detection and Tracking (TDT) program, in this chapter we start by describing the task of TDT in general. We then present an introduction to the TDT program, a brief history of it and its terminology, the TDT corpora, and an introduction to TDT evaluation methodologies. Then we describe topic tracking in more detail, and lexical chaining as a shallow natural processing approach to the tracking task. Finally, we discuss the 2002 Text Retrieval Conference (TREC 2002) filtering track task, and explain the reasons for using the 2002 TREC corpus and both TDT and TREC evaluation methodologies in our current work.

#### 2.1 Topic Detection and Tracking

Topic Detection and Tracking (TDT) research has been conducted under the DARPA Translingual Information Detection, Extraction, and Summarization (TIDES) program. The goal of the TDT program is to develop automatic technologies that discover and thread together topically related material from a variety of multi-lingual broadcast news media [1, 21].

Such automatic searching and organizing is potentially quite valuable for people who need timely and accurate access to large quantities of information. Systems built upon TDT could call the users' attention to new events and to new information about old events. By

examining one or two stories, a user could decide whether to pay attention to the rest of an evolving event. Similarly, a user could go to a large archive, find all the stories about a particular event, and learn how it evolved [1, 21].

The TDT program factors its research tasks into five areas [1, 21]:

Story Segmentation - the task of finding topically cohesive stories by segmenting the stream of data from a source

Topic Tracking - the task of finding stories about a user-designated topic

Topic Detection - the task of threading together stories that discuss the same event

New Event Detection - the task of detecting new topics

Link Detection - the task of deciding whether or not two stories are on the same topic

TDT research started with a pilot study in 1997 and has been through five open competitive evaluations, in 1998, 1999, 2000, 2001, and 2002. From the early focus of the pilot study, techniques have been extended into related areas, e.g., to information retrieval and speech recognition. TDT has by now evolved into a new domain, the finding of news-specific and event-focused stories, a refinement that should ideally improve system performance [1, 21].

## 2.2 TDT Terminology and Corpora

TDT research grew out of related research areas, such as information retrieval and text filtering. It differs from these areas mainly in that it has a separate set of terminologies and develops its own series of corpora for evaluations.

Crucial to TDT research are the definitions of *event* and *topic*. A TDT *event* is defined as “a specific thing that happens at a specific time and place, along with all necessary preconditions and unavoidable consequences” [2]. The TDT definition of a *topic* is ”a seminal event or activity, along with all directly related events and activities” [2].

In addition to the temporal nature and the locality pertaining to a topic in the above definitions, having an identifiable set of participants is a third characteristic of a topic.

3043.
Sri Lankan Gov't. vs. Tamil Rebels
Seminal Event
WHAT: New wave of violence breaks out between Tamil rebels and Sri Lankan government
WHERE: Sri Lankan
WHEN: late 1998
Topic Explication
Since 1983, more than 54,000 people have been killed in Sri Lanka's civil war between the majority Sinhalese who control the government and military, and the Liberation Tigers of Tamil Eelam, who are fighting for a separate homeland for minority Tamils in Sri Lanka's north and east. The fall of 1998 brought a new wave of violence and terrorism in this ongoing war. Although peace talks looked likely in late 1998, the fighting had begun again by January 1998.
On topic: Any stories covering acts of violence or terrorism in this conflict; investigations by external organizations (like Amnesty International); peace negotiations between the opposing sides.
Rule of Interpretation Rule 6: Ongoing Violence or War
Related Article: VOA 1998 1015.0600.0290, APW 1998 1110.0220
More examples: Yes, Brief

Figure 2.1: An example topic definition

To illustrate these three characteristics of a TDT topic, we quote a TDT-3 topic definition from [2] in Figure 2.1:

TDT-3 in Figure 2.1 is one of a series of corpora developed to evaluate the TDT systems. TDT 1997 used the Reuters corpus. The evaluations since 1998 used TDT-2, TDT-3, and TDT-4 corpora which have been developed by the Linguistic Data Consortium (LDC).

## 2.3 TDT Evaluation Methodology

The terminology and formulas in this section can be found in [3].

TDT evaluation methodology uses two techniques to represent system performance: the detection cost function ( $C_{Det}$ ) and the decision error tradeoff (DET) curve, which are based on missed detections and false alarms. For each of the five TDT tasks, a target story can be correctly detected as a target, or a story can be missed. The later case is called a missed detection. Similarly, a non-target can be correctly determined to be a non-target, or it can be falsely detected in which case the error is called a false alarm. *Missed detection* and *false alarm* can be understood better with the TDT contingency table shown in Figure 2.2

System Response		Target	Non-Target
	YES (a Target)	Correct	Flase Alarm
	No (Not a Target)	Missed Detection	Correct

Figure 2.2: TDT Contingency Table

Given the contingency table in Figure 2.2, TDT research defines the evaluation formulas listed in Table 2.1.

$$C_{Det} = C_{Miss} * P_{Miss} * P_{Target} + C_{Fa} * P_{Fa} * (1 - P_{Target}) \quad (2.1)$$

$$P_{Miss} = \#ofMissedDetections / \#ofTargets \quad (2.2)$$

$$P_{Fa} = \#ofFalseAlarms / \#ofNon - Targets \quad (2.3)$$

Table 2.1: table title

In formula 2.1  $C_{Miss}$  and  $C_{Fa}$  are the costs of a missed detection and of a false alarm respectively. They are pre-specified in the application. In formula 2.2,  $P_{Miss}$  is the probability of a missed detection.  $P_{Fa}$  in formula 2.3 is the probability of a false alarm. Both  $P_{Miss}$  and  $P_{Fa}$  are determined by the evaluation results.  $P_{Target}$  is the *a priori* probability of finding a



target as specified by the application.

The DET curve visualizes the tradeoff between the probabilities of missed detections and false alarms using the distribution of decision scores. A system that performs well is supposed to have small  $P_{Miss}$  and  $P_{Fa}$  values. In other words, the DET curve for a good system should be close to the origin.

## 2.4 Topic Tracking

In present work, we are concerned with the task of topic tracking. As defined in section 2.1, topic tracking systems detect stories that discuss a previously defined topic. As all TDT tasks grow out of related research in information retrieval, the tracking task is similar in spirit to information retrieval’s filtering task. We will discuss the similarities between a tracking system and a filtering system below in section 2.5.

A tracking system proceeds as follows [1, 21]: The system is given a small number of stories (from one to sixteen, typically from one to four) that are known to be on the same topic. In TDT terminology, these are the  $N_t$  stories. Then the system tries to find all other stories on this topic in the stream of arriving news.

Algorithmically, a TDT tracking system works as follows:

1. Extract a set of features from the training stories that differentiate them from the much larger set of stories in the past.
2. Compare a new story to the topic features and if it matches sufficiently, declare it to be on topic.

TDT research requires a tracking system to have the three following design features [3]:

- The system is usually given multiple topics and tracks all topics independently.
- The system normalizes the decision scores across topics.
- The system is required to be multi-lingual. That is, the system is supposed to track topics in all languages within the corpus regardless of all training/test pairs.

In five previous TDT evaluations, systems have been built for the tracking task. Most of the systems rely on statistical approaches. Tracking effectiveness of these statistical systems in recent evaluations has been shown to be fairly good. Carthy in the TDT 2001 evaluation proposed the first tracking system that was not based only on a statistical approach. Carthy added to his tracking system a lexical chaining process, a shallow natural language processing technique. The next section outlines the way Carthy’s tracking system uses lexical chains.

#### 2.4.1 Tracking using Lexical Chaining

Lexical chaining is the process of finding a sequence of semantically related words spanning documents. Lexical chaining falls into the scope of shallow natural language processing (NLP). Unlike full NLP approaches, shallow NLP does not require complete parsing and extraction of semantic information. Instead, shallow NLP is marked with two characteristics [33]. First, shallow NLP analyzes and quantifies textual contexts in documents; Second, shallow NLP does not extend beyond the scope of part of speech tagging and morphological analysis.

Based on Halliday’s linguistic theories [18] on cohesion in English, Hirst and Morris [31] proposed the concept of lexical chains in 1991. Since then several lexical chaining algorithms have been developed and applied to hypertext linking, text summarizing, text segmentation,

information retrieval, malapropism detection, and multimedia indexing. Carthy applied lexical chaining to topic tracking and presented his tracking algorithm at the TDT 2001 evaluation. We can summarize Carthy’s tracking algorithm as follows [11]:

Associate the topic to be tracked with one or more ( $N_t$ ) stories as specified in the TDT tests. Then the tracking system proceeds through the steps listed in Figure 2.3

1. Compute the tracking set of lexical chains for the current event from the  $N_t$  stories for this event.
2. Construct a tracking query for the current event from the  $N_t$  stories for this event.
3. Compare each new document in the document stream to the tracking descriptor (comprising the tracking set and tracking query).
4. If the new document is sufficiently similar to the tracking descriptor then flag it as tracking that event.
5. Repeat steps 1 to 3 for all  $N_t$  values.
6. Repeat steps 1 to 4 for all events to be tracked.

Figure 2.3: Carthy’s Tracking System

## 2.5 TREC 2002 Filtering Track

As stated in [1] and mentioned in section 2.4, the TDT tracking task is similar in spirit to the filtering task of Text Retrieval Conference (TREC). In this section we briefly describe the 2002 TREC filtering task, evaluation methodology, similarities between the TDT tracking and TREC 2002 TREC filtering task, and reasons for using the TREC 2002 corpus and evaluation methodology in our work.

### 2.5.1 Definitions and Evaluation

TREC 2002 defines the filtering track task: “Given a topic description and some examples of relevant documents, build a filtering profile which selects the most relevant examples from

an incoming stream of documents [34]”. The filtering track task consists of three subtasks: adaptive filtering, batch filtering, and routing.

Algorithmically, adaptive filtering works as follows:

The system starts with a set of topics and a document stream, and a small number of examples of relevant documents. For each topic, it must create an initial filtering profile and then analyze the incoming documents and make a binary decision to retrieve or not to retrieve each document on a profile by profile basis. If a document is retrieved for a given profile and a relevance judgment exists for that document/topic pair, this judgment is provided to the system and the information can be used to update the profile.

TREC 2002 filtering task uses the Reuters Corpus Volume 1 for evaluation purposes.

The definition of a filtering track *topic* is illustrated in Figure 2.4.

```

<top>

<num> Number: R104

<title> Rescue of kidnapped children

<desc> Description

        Identify a kidnapping of a child or children when the child or children
        have been rescued or released.

<narr> Narrative:

        Documents discussing abducted or kidnapped children are relevant.
        Documents referring to abuse of children without reference to kid-
        napping or abduction are irrelevant. Cases of kidnapping where some
        children are murdered or not found while others are rescued are re-
        levant.

</top>

```

Figure 2.4: Topic #104 in Reuters Corpus

TREC 2002 filtering track defines the contingency table shown in Figure 2.5.

	Relevant	Not Relevant
Retrieved	$R+ / A$	$N+ / B$
Not Retrieved	$R- / C$	$N- / D$

Figure 2.5: TREC Contingency Table

Table 2.2 lists the TREC 2002 evaluation functions based on the contingency table in in Figure 2.5.

- Utility Measures

$$LinearUtility = A * R+ + B * N+ + C * R- + D * N- \quad (2.4)$$

( $A = 2, B = -1, C = D = 0$ ):

$$T11U = 2 * R+ - N+ \quad (2.5)$$

$$MaxU = 2 * (Totalrelevant) \quad (2.6)$$

Normalized utility:

$$T11NU = \frac{T11U}{MaxU} \quad (2.7)$$

$$T11SU = \frac{\max(T11NU, MinNu) - MinNu}{1 - MinNu} \quad (2.8)$$

- F-beta Measure if  $R+ = N+ = 0$

$$T11F = 0 \quad (2.9)$$

otherwise

$$\frac{1.25 * R+}{0.25 * R- + N+ + 1.25 * R+} \quad (2.10)$$

- Precision and Recall
- Average Uninterpolated Precision

Table 2.2: TREC 2002 Evaluation Functions

According to the guidelines in [34], the larger the utility scores a filtering system achieves, the better that system performs for a query. Similarly, the F-beta measure ranges from zero to one, with zero being bad and 1 being good.

## 2.6 TDT and TREC Tracking

As Allen states in [1], the TDT tracking is in spirit similar to TREC filtering tracking.

We can see this in the design issues of both tracking tasks. Section 2.4 quotes the design issues for TDT tracking. To repeat them here, a TDT tracking system is supposed to handle topics independently, normalize scores across topics, and process multi-lingual corpora. The guidelines for TREC 2002 adaptive filtering quoted in the beginning of this section state that a filtering system must process each topic independently. The utility functions in Table 2.2 above are normalized across topics. The TREC overview in [40] states that systems are also built to retrieve documents across multiple languages.

Given that TDT tracking and TREC tracking are similar tasks, the former differs from the latter in two ways.

First, TDT tracking is more oriented toward tracking event-based news stories. Hence, TDT tracking emphasizes the *what*, *where*, *when*, *who* and *why* information of a topic as illustrated in a definition from TDT-3 in Figure 2.1. But TREC tracking does not seem to explicitly highlight the *what*, *where*, *when*, *who* and *why* information of a topic as shown in the definition of TREC topic #104 in Figure 2.4.

Second, the evaluation methodologies are somewhat different. The TDT cost function  $C_{det}$  differs from the TREC utility and F-beta measures.

However, TDT DET curves are relevant to TREC precision-recall measures because  $Precision = 1 - P_{miss}$  and  $Recall = 1 - P_{fa}$ .

In current work, we use both TDT and TREC evaluation methodologies for a couple of reasons. First, TDT tracking and TREC tracking are similar tasks as explained above.

Second, we have access to Reuters Corpus Volume I since Dr. Mihalcea has participated TREC 2002 but not to any of the TDT corpora. Also, Reuters Corpus Volume I was used in the 1997 TDT pilot study. Third, the DET curve, the important measure of TDT tracking systems, is relevant to the precision-recall curve. Fourth, all TDT and TREC evaluation statistics can be computed using the same set of data because they are all derived from the TDT and TREC contingency tables, both of which say the same things.

## CHAPTER 3

### ISSUES WITH EARLIER LEXICAL-CHAINING ALGORITHMS

At the core of Carthy’s tracking system outlined in Figure 2.3 chapter 2 is the chaining process. This process also forms the kernel of several lexical-chaining-based information processing systems, such as hypertext linking, text summarizing, text segmentation, information retrieval, and multimedia indexing. This chapter describes the linguistic background for lexical cohesion, defines lexical chains, and summarizes the features of and lessons learned from three of the systems mentioned above.

#### 3.1 Meaning of Text and Lexical Cohesion

Recall that the task of topic tracking is to determine if two texts are similar, that is, to determine if the two express similar ideas. To build an automated system to extract meaning from a text does not appear to be an easy task. Fortunately, Halliday and Hasan’s description of textual meaning through lexical cohesion among sentences makes the job computationally feasible.

According to Halliday and Hasan, a text is more than a string of sentences. It is a semantic unit, which is a unit of meaning in context. Being a semantic unit, a text is realized in the form of sentences. A set of related sentences is the embodiment or realization of a text. So the expression of the semantic unity of the text lies in the cohesion among the sentences of which it is composed [18].

Halliday and Hasan [18] define cohesion as a semantic relation realized through the lexical



or the grammatical system. The dependency relationship between words makes up the lexical cohesion system. The cohesive effect of this system is obtained by the selection of vocabulary. Halliday and Hasan list two classes of lexical dependency relationship - reiteration and collocation.

Reiteration is the repetition of a previous semantic concept. It involves the following three types of repetition techniques:

- The use of a general noun to refer back to a lexical item. For example: *What shall I do with all this **crockery**? Leave the **stuff** there; someone'll come and put it away.*
- The repetition of a lexical form with/without identity of reference. Consider two sentences [31]: *Mary bit into a peach. Unfortunately, the peach wasn't ripe.*
- The use of a synonym, near-synonym, or superordinate. For example [31]: *Henry's bought himself a new **Jaguar**. He practically lives in the **car**.*

Collocation is achieved through the association of lexical items that regularly co-occur. Halliday and Hasan illustrate two types of collocation: systematic semantic relationship and non-systematic semantic relationship [18, 31].

- Systematic semantic relationship includes antonyms, members of an ordered set such as *one, two, three*, members of an unordered set such as *black, white, red*, and part-whole relationships like *eyes, mouth, face*.
- Non-systematic semantic relationships exist between words that tend to share the same lexical environments. Words tend to occur in similar environment because they describe

things that coexist in similar situations or contexts in the world. An environment-specific example of this relationship can be found in the set { *post office*, *stamps*, *envelope*, *express mail* } from the post office scenario.

When two lexical items appear in similar contexts due to one of the abovementioned semantic relationships, they will generate cohesion in the same sentence or neighboring sentences. This cohesive effect can cross the limit of two words. Long cohesive chains can then be built out of successive sentences or even for the whole document.

From Halliday and Hasan’s explanation of the meaning of text that arises through lexical cohesion, we may conclude that if a topic-tracking system can extract lexical items related by the cohesion techniques listed above from a text, then it can zero in on the components that contribute to the meaning of the text. Hirst and Morris develop this approach further in their work as described in the next section.

### 3.2 Definition of Lexical Chain

Halliday and Hasan’s linguistic theories provide a platform for Hirst and Morris, who in 1991 formally defined lexical chains and proposed an algorithm to construct them computationally [31].

They define a lexical chain as a sequence of semantically related words in a text, which creates a context and contributes to the continuity of meaning. A lexical chain may span short distances (adjacent words or sentences) or long ones (an entire text). A chain is independent of the grammatical structure of the text; in effect it is a list of words that captures a portion of the cohesive structure of the text [31].

Hirst and Morris also describe two important features of lexical chains. With the elaboration of later researchers, we can summarize these features as follows:

A lexical chain can provide a context for the resolution of an ambiguous term and enable identification of the concept that the term represents. When a word is added to a chain, its ambiguity may be resolved. The sense of this word is therefore determined by rejecting other candidate senses that would be valid for other lexical chains. This method is called incremental word sense disambiguation because each time a word is added to a chain, it may help clarify the sense of the other words of the chain [31, 12, 11].

Lexical chains “provide a clue for the determination of coherence and discourse structure, and hence the larger meaning of the text” [31]. Hirst and Morris explain this “clue” for “the larger meaning of the text” through the relationship between lexical chains and linguistic structure. Linguistic structure is the segmentation of discourse into groups of sentences, each fulfilling a distinct role in the discourse. Lexical chains indicate the linguistic segmentation. When a lexical chain ends, there is a tendency for linguistic segment to end, as the lexical chains tend to indicate the typicality of segments. If a new lexical chain begins, this is an indication or clue that a new segment has begun, hence a new idea. If an old chain is returned to again (a chain return), it is a strong indication that a previous segment/idea is being returned to [31].

Discussions in sections 3.1 and 3.2 provide for the linguistic background for a topic tracking system that is based on lexical chains because they contribute to the cohesion and meaning of a text.

### 3.3 A Generic Chaining Algorithm

Hirst and Morris proposed a lexical chaining algorithm. They hand-computed lexical chains using *Roget's International Thesaurus*. Since 1995 several automated lexical chainers have been built using the WordNet database. These chainers more or less follow a generic chaining algorithm similar to Hirst and Morris'. The algorithm is stated as follows [31]:

1. Select a set of candidate words from the text;
2. For each of the candidate words, find an appropriate chain to receive the candidate word, relying on a relatedness criterion among members of the chains and the candidate words;
3. If such a receiving chain is found, insert the candidate word in this chain and update it accordingly; otherwise, create a new chain.

Even with their hand-computation, using Roget's Thesaurus, Hirst and Harris found that the thesaurus failed to confirm an intuitive lexical chain due to three missing knowledge sources.

- General semantic relations between words of similar “feeling”.
- situational knowledge.
- specific proper names.

These problems appeared when Hirst and Morris computed lexical chains with Roget's Thesaurus. Would similar problems occur in later lexical chainers using the WordNet

database? Section 3.4 illustrate how WordNet makes computing the chains feasible and discusses two potential problems in using WordNet to find the relatedness between lexical items.

### 3.4 WordNet and the Chain Computing

WordNet is an electronic lexical database developed by psycholinguists led by George Miller at Princeton University [30]. The WordNet project is inspired by psychological theories of human knowledge of the mental lexicon. WordNet groups English nouns, verbs, adjectives and adverbs into synonym sets, the synsets. Each synset represents a lexical concept. The synsets are organized by four semantic relations, namely, synonymy, antonymy, hyponymy/hypernymy and meronymy/holonymy, which form the mental lexicon.

A weakened definition of synonymy is given by Miller in [30]. It relies on a context we see in the following: “two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value”. Instead of giving antonymy a formal definition, Miller states that antonymy is key to organizing the adjectives and adverbs in WordNet. The hyponymy/hypernymy relation corresponds to the subordination/superordination relation, which forms a hierarchical semantic structure. The meronymy/holonymy relation stands for the HAS A/ISPART OF relation.

#### 3.4.1 Computing Relatedness

Due to the organization based on the above four semantic relations, WordNet can be used to find the synonyms, antonyms, hyponyms/hypernyms, and meronyms/holonyms for each

noun, adjective, verb or adverb in a document. Hence, WordNet provides a method for finding the relatedness between any two lexical items in the document. In terms of Halliday and Hasan's lexical dependency relationship, WordNet provides a clue for lexical and textual cohesion by returning synonyms, antonyms, hyponyms/hypernyms, and meronyms/holonyms. WordNet makes computing lexical chains possible. To illustrate this, consider the results WordNet returns for the sense # 1 for words *stamp* and *postage* respectively in Figures 3.1 and 3.2.

1. {05563469} postage, postage stamp, stamp --  
 (the token that postal fees have been paid)

Figure 3.1: Sense #1 of *stamp*

1. {05563469} postage, postage stamp, stamp --  
 (the token that postal fees have been paid)

Figure 3.2: Sense #1 of *postage*

Figures 3.1 and 3.2 show that *stamp* and *postage* belong to the same synset and hence are related within the same lexical concept.

### 3.4.2 Potential Problems

However, there exist at least two uncertainties with using WordNet to find the relatedness between lexical items.

One of the uncertainties is the designers' original plan not to include proper nouns. Topic tracking is intended to track event-based news stories where the time and location of the events play an important role. Although WordNet has included some proper nouns, if

WordNet does not provides information for all proper nouns, a tracking system completely relying on WordNet for lexical relatedness would not be optimal.

The other uncertainty is that WordNet does not seem to relate lexical items associated in the non-systematic semantic relationship. Consider the pair *post office* and *stamp* from the set of the post office scenario. If we look up all the synonyms, antonyms, hypernyms, hyponyms, myronyms and holonyms of *post office*, we would not be able to find that the synset numbers of *stamp* or the word *stamp* itself appears in the relation sets returned by WordNet. This causes WordNet potentially not to be fully reliable for the systems depending on WordNet.

The rest of this chapter analyzes some of the chaining algorithms and issues related to using WordNet as the knowledge database.

### 3.5 Issues with Earlier Chainers

As mentioned earlier, there are several applications of lexical chaining. This section briefly describes Stairmand's chainer for information retrieval, and hypertext linking, and Kazman and Al-Halimi's chainer for multimedia indexing.

#### 3.5.1 Stairmand's Chainer for IR

Stairmand built the first automated lexical chainer in 1994, using the WordNet database. The chainer was developed for information retrieval purposes. His algorithm works as follows [38]:

1. Tag the input document. Collect all the lexical items labeled as nouns, adjectives, and verbs. These are the candidate words.

2. Assign a paragraph number to each candidate word.
3. Using WordNet, for each of the candidate words, find its synonyms, subordinates, immediate superordinates, and meronyms. These are the expansion terms for the candidate.
4. Find all possible links among the candidate words using their expansion terms.
5. The candidate words that are linked together from step 4 form lexical chains.

The algorithm also calculates the density of each chain. It keeps the high-density chains and rejects the ones with low-density. While this algorithm follows the outline of the generic algorithm, it does not try to handle compound words, proper nouns, and expressions that are not in WordNet.

### 3.5.2 Greedy Chainers

While Stairmand’s algorithm is not greedy, Green, St-Onge, and Kazman and Al-Halimi all take a greedy approach, in the sense that they compute the relationships between words by considering certain kinds of paths between synsets in the WordNet graph and selecting the immediate sense from the many senses of a candidate word that meets the relatedness criteria. This early decision drops the rest of the senses of the word. Since St-Onge’s algorithm is representative of the greedy chainers, the next section analyzes St-Onge’s chainer for malapropism in detail.



### 3.6 St-Onge’s Chainer

This section compares the chains resulted from St-Onge’s chainer against intuitive chains and analyzes the comparisons.

#### 3.6.1 The Algorithm

St-Onge’s algorithm can be summarized as follows:

1. Keep a stop word list which consists of closed-class words, and the vague high-frequency words that tend to weaken chains, e.g. *one*, *two*, *dozen*, *little*, *might*.
2. Input the words one by one. Select the nouns and noun bases of verbs as the candidate words.
3. Combine the input words into compound expressions. If a compound expression exists in WordNet, select it as a candidate word.
4. Compute the four types of relations between words. An extra-strong relation relies on word repetition. A strong relation has three subcategories. The first one relates to the common synsets between two words. The second one corresponds to the antonyms between a synset of each word. The last category of the strong-relation refers to the IS-A relation between two words. A medium-strength relation between two words relies on a special path between a synset of each word.
5. To find the chain corresponding to a given candidate word, extra-strong relations have a higher priority than strong relations, and both of them have higher priority than medium-strong relations.

6. Push each candidate word in a sentence into a queue.
7. For each candidate in the queue, find its extra-strong relation in the chain. A search ends as soon as the extra-strong relation is found. Otherwise, the strong and medium-strong relations are searched in turn.

This algorithm is illustrated with an example from Barzilay in section 4.1.1 of chapter 4.

### 3.6.2 A Sample Text and its Intuitively Natural Chains

Consider the following sample text and result chains taken from [37].

*We suppose a very long train traveling along the rails with the constant velocity  $v$  and in the direction indicated in Figure 1. People traveling in this train will with advantage use the train as a rigid reference body; they regard all events in reference to the train. Then every event which takes place along the line also takes place at a particular point of the train. Also, the definition of simultaneity can be given relative to the train in exactly the same way as with respect to the embankment.*

According to St-Onge in [37], three chains can be built intuitively from the above sample text.

Chain 1 - {train, rails, train, train, train, line, point, train, train, embankment}

Chain 2 - {traveling, velocity, direction, traveling}

Chain 3 - {reference-body, reference}

The context created by Chain 1 enables the identification of the appropriate meaning of the word *line*. Word sense disambiguation is, in fact, an important application for lexical

chaining.

St-Onge’s greedy algorithm generates the following nine chains [37]:

Chain 1 - {train, velocity, direction, train, train, train, advantage, reference, reference\_to,  
train, train, respect\_to, simultaneity}

Chain 2 - {traveling, traveling}

Chain 3 - {line, rails}

Chain 4 - {given, constant}

Chain 5 - {body, people, figure}

Chain 6 - {point, particular, regard}

Chain 7 - {place, place, event, events}

Chain 8 - {definition}

Chain 9 - {embankment}

To his surprise, St-Onge found that his chains are far from being perfect. Here are some of his observations:

Observation 1: The algorithm fails to include *rails* in Chain 1, which according to out intuition should be included. St-Onge claims that this is because the distance between *train* and *rails* in WordNet is too large.

Observation 2: *Simultaneity* is added to Chain 1. The algorithm wrongly disambiguated *simultaneity* by associating its sense of *happening or existing or done at the same time* with

one of the senses of *train* - *a sequentially ordered set of things*. But the sense *a line of railway cars coupled together and drawn by a locomotive* of *train* is expected here.

Observation 3: Chain 5 also puts together words that are not semantically related. The words *people* and *figure* in the sample text are not successfully disambiguated. Obviously, in the sample text, *figure* refers to a *diagram or picture* instead of a *human body* or *physical body* of *people*. The latter sense is apparently how St-Onge’s chainer disambiguated *figure*.

### 3.6.3 Over and Under Chaining

St-Onge came up with the terms of over-chaining and under-chaining to explain the above three observations. According to him, under-chaining is the inability to link a pair of related words, i.e. in Observation 1. Similarly, over-chaining is the linking of two poorly related words, as in Observations 2 and 3.

To explain these, St-Onge gives the following reasons:

- (a) An inadequacy of WordNet’s set relations. For instance, *child care* and *school* cannot be related by using WordNet’s set of relations;
- (b) A lack of connections in Wordnet. For instance, WordNet does have a proper set of relations to link *beef stew* and *beef* with a single relation (substance meronym/holonym), but no such link exists in WordNet’s graph.
- (c) A lack of consistency in the semantic proximity expressed by WordNet’s links. For example, in WordNet’s graph, the shortest path between *stew* and *steak* has 6 links, while the shortest path between *Australian* and *millionaire* has 4 links.
- (d) A poor algorithm for chaining.

(a) and (b) correspond to the second potential problem we brought up in 3.4.2 about using WordNet as the only knowledge database. (c) shows that the inconsistent distance between words' senses could also cause under- and over-chaining problems.

### 3.6.4 Suggestion for Improvement

Given the under-chaining and over-chaining problems with his algorithm and their causes, St-Onge has looked for a more accurate way to compute semantic proximity. He suggests combining another knowledge database, like Roget's Thesaurus, with WordNet, in order to get the best out of both knowledge databases. But presently, this is not feasible, because the combination would require matching corresponding word senses of each entry in the two knowledge databases.

Thus St-Onge has apparently not yet developed any improvements for his algorithm. In addition, St-Onge does not mention how to handle proper nouns.

## 3.7 Summary of the Issues with Early Chaining Algorithms

Two types of problems can be identified in the above algorithms.

The first type has to do with the chaining algorithms themselves.

The second type, which includes four subproblems, is inherent in the organization of the WordNet database itself.

- There are few links between noun, verb, adjective and adverb hierarchies, while a complete chaining system requires connections between the three parts of speech. This is why most chainers except for Stairmand's use only the noun database of WordNet.

- Some important classes of words, mostly proper nouns, are left out of the noun database.
- WordNet does not provide information to connect lexical items that are linked by non-systematic semantic relationships.
- As Butanisky pointed out in [20], the fine-grainedness of WordNet senses causes unexpected results for chaining algorithms.

Chapter 4 discusses some chaining algorithms that try to overcome some of the problems listed above.

## CHAPTER 4

### IMPROVING THE CHAINING PROCESS

Chapter 3 outlines the work that has to be done in order to improve previous lexical chaining systems. Specifically, two types of work need to be done - improving previous chaining algorithms and enhancing the WordNet database. Efforts have been made to tackle the problems arising from the chaining algorithms and those inherent in the WordNet database. Barzilay, Carthy, and Silber and McCoy developed chainers to optimize the chaining process. We call these “later chainers”. Mihalcea and Moldovan built a coarse-grained WordNet, which is supposed to shorten the distance between related synsets and create more connections between the lexical items [28]. Magnini et. al. proposed the domain-label-based approach to optimize their word-sense disambiguation system [24]. In sections 4.1, 4.3, and 4.2, we will present Barzilay, Silber and McCoy, and Carthy’s chaining systems, in that order. In the next chapter, we present our own proposal for a chaining algorithm, one based on semantic domains.

#### 4.1 Barzilay’s Dynamic Chaining Algorithm

Barzilay’s dynamic chaining algorithm is one of the three later chainers which aim to improve the lexical chaining process using WordNet. Barzilay had studied earlier chaining algorithms carefully before she proposed her dynamic algorithm.

#### 4.1.1 Comments on St-Onge’s Algorithm

Barzilay holds that greedy algorithms, by selecting the immediate sense from the many senses of a candidate word that meets the relatedness criterion, lose other senses of the candidate word. Hence, the greedy disambiguation process leads to under- and over-chaining. Barzilay illustrates this by computing the chain with the greedy algorithm [5] for the following passage:

*Mr. Kenny is the person that invented an anesthetic machine which uses micro-computers to control the rate at which an anesthetic is pumped into the blood. Such machines are nothing new. But his device uses two micro-computers to achieve much closer monitoring of the pump feeding the anesthetic into the patient.*

When St-Onge’s algorithm runs, it first creates the chain for the word *Mr.*, holding one sense:

Chain-1:  $\{\{Mr., \text{sense} - (\text{mister}, Mr.)\}\}$ . *Mr.* belongs to one synset, so it is disambiguated from the beginning.

When candidate word *person* is processed. It is related to the above chain in the sense of *a human being*, so chain-1 now contains two entries:

Chain-1:  $\{\{Mr., \text{sense} - (\text{mister}, Mr.)\}, \{person, \text{sense} - (\text{person}, \text{individual}, \text{someone}, \text{man}, \text{mortal}, \text{human}, \text{soul})\}\}$

Then the algorithm processes the word *machine*, it relates it to chain-1, because *machine* in the first WordNet sense (an efficient person) is a holonym of *person* in the chosen sense. According to Barzilay in [5], *machine* here is not correctly disambiguated, even though after this first occurrence of *machine*, there is strong evidence supporting the selection of its more common sense: “micro-computer, device and pump”, all pointing to its correct sense in this



context.

Hence, Barzilay concludes that the disambiguation process cannot be achieved by a greedy decision.

#### 4.1.2 Barzilay's Algorithm in Brief

Barzilay's algorithm can be summarized in terms of the first two phases of the generic chaining process.

In the candidate-selecting phase, the algorithm follows the following steps:

1. Segment the input file, using Hearst's algorithm.
2. For each segment, find collocations by checking to see if pairs of sequential nouns exist in WordNet.
3. Tokenize the words in the segment.
4. Tag each word with its corresponding part-of-speech using Brill's tagger.
5. Select compound nouns or head nouns of the compound nouns if they do not exist in WordNet.
6. Select nouns resulted from steps 3, 4 and 5.

The chaining phase is dynamic. Specifically, the decision about the right sense of a word is delayed. Several exclusive chains coexist at the same time for a candidate word before the chain with the highest score is picked.

### 4.1.3 Lessons Learned

Results from Barzilay’s experiments show that the dynamic algorithm outperforms the greedy algorithm. While Barzilay does not comment on her algorithm, she lists the lessons learned from the experiments. The most relevant lesson for us is that: cleaning WordNet of rare senses, and aggregation of similar senses can dramatically improve performance.

## 4.2 Silber and McCoy’s Linear Algorithm

While Barzilay sufficiently overcomes the under-chaining and over-chaining problems of greedy algorithms, Silber and McCoy come up with a linear algorithm to improve the running time of Barzilay’s dynamic algorithm. They obtain the improved performance by representing the sense graph as a sense array, whereas Barzilay uses actual graphs to hold word senses. To index their sense array, Silber and McCoy re-indexed the WordNet database.

### 4.2.1 The Linear Algorithm

In [36], Silber and McCoy present their algorithm as follows:

1. Tag the corpus, using Brill’s Tagger;
2. For each noun in the source document, form all possible lexical chains by looking up all relational information including synonyms, hyponyms, hypernyms, and siblings. This information is stored in an array indexed on the index position of the word from WordNet for constant time retrieval;
3. For each noun in the source document, use the information collected by the previous step to insert the word in each *meta-chain*. A *meta chain* represents all possible chains

whose beginning word has a given sense number. Meta-chains are stored by sense number. The Sense numbers are now zero-based due to their re-indexing of WordNet.

According to Silber and McCoy, a *meta-chain* is a representation of every possible lexical chain that can be computed starting with a word of a given sense.

#### 4.2.2 Modifying the WordNet Noun Database

Silber and McCoy re-indexed the WordNet noun database by line number as opposed to file position. The file was saved in a binary indexed format. The database access tools were then rewritten to take advantage of this new structure. The result of this work is that accesses to the WordNet noun database can be accomplished in an order of magnitude faster than with the original implementation.

#### 4.2.3 Advantages of the Linear Algorithm

The linear algorithm has the following advantages:

- Obviously, the complexity is linear;
- The algorithm is also linear in space requirements. Due to this, Silber and McCoy can consider all possible chains. Barzilay had to prune interpretations (and thus chains) which did not seem promising;
- The algorithm allows Silber and McCoy to analyze the importance of segmentation. While Barzilay has to use segmentation to reduce the complexity of the problem of extracting chains, Silber and McCoy do not need to segment the corpus if it is not too long.

### 4.3 Carthy's Chainer

Carthy proposes the first topic tracking system based on lexical chaining for the 2001 topic detection and tracking (TDT) evaluation project. Carthy's chainer is non-greedy and similar to Barzilay's. The former differs in that the chaining process is not done with graphs. We quote Carthy's chaining algorithm in Figure 4.1. We will come back on this chainer in section 6.1.1.

1. For the  $i$ th candidate word from the tagged story, find the synset numbers of all its senses in WordNet.
2. If this candidate word is tagged as a proper noun, insert it into the Proper Noun chain.
3. Else, take the  $i$ th term in the story and using WordNet, generate the set called Neighbor  $i$  of its related synsets using the hyponym/hypernym and meronym/holonym relationships.
4. For each other term in the document, if it is a member of the set Neighbor  $i$  then add its synset identifier to the lexical chain for term  $i$ . The location of the chain element in the document stream is also stored.
5. If the lexical chain contains 3 or more elements then store the chain in a chain index file
6. Repeat steps 1, 2 and 3 for all terms in the story.

Figure 4.1: Carthy's Chaining Algorithm

In the paper submitted to the 2001 TDT evaluation project, Carthy does not provide results from his experiments with his chainer. We cannot report Carthy's results here.

### 4.4 Modifying the WordNet Database

Barzilay somehow resolves the under- and over-chaining problem using a dynamic chainer. Silber and McCoy improve the time and space complexities of the chaining process by modifying the offset numbers in the noun database. In addition to Carthy's chainer, the chaining

algorithms in this chapter are all focused on improving the chain algorithm per se. Hence, it seems that little work has been done in lexical chaining to improve the issues inherent in the organization of the WordNet database, so that better chaining performance can be attained. This appears to be a difficult task.

In the field of word sense disambiguation, where researchers also rely on the WordNet database, work has been done to bring some change. The issues inherent from the WordNet database, as outlined in section 3.7, are missing the representation of the non-systematic semantic relation, the long distance between semantically related items, the fine-granularity of senses and the lack of connection between the database of different parts of speech. Mihailescu's EZ. WordNet and Magnini's domain labels are two of the techniques that are intended to deal with the above issues with the WordNet database.

Mihailescu's EZ. WordNet [28] is built to shorten the distance between semantically related lexical items by two procedures, as follows:

- Combine synsets similar in meaning.
- Drop rare synsets.

Magnini added the "WordNet Domains", as an extension of WordNet 1.6, to achieve better performance in his word-sense disambiguation system. The domains built by Magnini enhance his WordNet-based disambiguation system in three ways [24].

First, a domain may group together lexical items from databases from different parts of speech which creates links between different databases. The organization of the databases in WordNet does not assume connections between the noun, adjective, verb and adverb files. As an example, the domain *MEDICINE* brings into a group the nouns *doctor* and *hospital* and

the verb *operate*, whereas searches in the original WordNet will not return such connection between *operate* and *doctor* or *operate* and *hospital*.

Second, a domain may group together lexical items that are related by non-systematic semantic relationships. In [37], St-Onge gives the pair { *stew*, *steak* } to illustrate that WordNet does not help to find any relatedness between them. However, using Magnini's domain labels, we can find that the synset number *06167458* of *stew* and the synset number *06228083* of *steak* both belong to the domain *gastronomy*. Hence *stew* and *steak* are related. Figure 4.2 shows another set of words grouped by the domain *sport*, which are otherwise not related simply using WordNet itself.

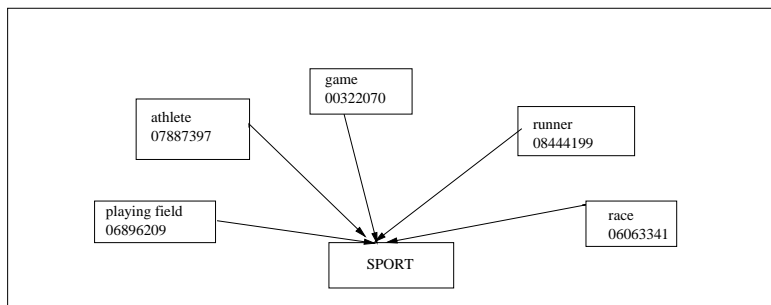


Figure 4.2: Relatedness by Semantic Domain

Third, a domain may combine similar senses of one lexical item into one cluster and thus reduce the synsets in WordNet. This is similar to Mihalcea's first procedure stated above. The result is the reduced polysemy in WordNet. In other words, Magnini's extension leads to a coarser-grained WordNet.

The three features of Magnini domains seem to solve the problems inherent in the inner structure of the WordNet database.

Magnini computes about 200 domain labels for the synsets in WordNet 1.6. Since the

original domains were defined on WordNet 1.6, we had to compute our own mapping from WordNet 1.6 to WordNet 1.7.

## CHAPTER 5

### CHAINING WITH SEMANTIC DOMAINS

This chapter illustrates the feasibility of building lexical chains with semantic domains and presents our domain-based lexical chainer.

#### 5.1 Semantic Domains and Lexical Chains

A semantic domain is a relational structure of concepts which are brought together by a theory or a coherent set of beliefs. Semantic domains refer to the traditionally studied “semantic fields”, which are sets of conceptually related lexical items. Hence, the lexical items grouped together under the same domain are constrained by a common semantic relation. For example, the set of words in Figure 4.2 involve the same semantic concept *sport*. According to the definition in section 3.2, “a lexical chain is a sequence of semantically related words in a text, which creates a context and contributes to the continuity of meaning”. Since the lexical items in Figure 4.2 are semantically related to the concept of *sport*, they form a lexical chain. Therefore it is possible to compute the lexical chains in a document according to the semantic domains the lexical items belong to.

A synonym set in the WordNet database corresponds to a group of lexical items under the same domain or semantic field. In this sense, Magnini’s semantic domains are similar to WordNet synsets, in that they both refer to conceptually related lexical items. However, Magnini’s semantic domains differ, in that these domains connect the WordNet synsets that are not originally connected in the WordNet database. We can call each of Magnini’s



semantic domains “a domain of domains”, with “domains” being the synsets from WordNet.

## 5.2 Domain Lexical Chainer

We have illustrated in sections 4.4 and 5.1 that it is not only possible to build chains with semantic domains but also to resolve the issues inherent in the organization of the WordNet database. Now we are ready to present our tentative domain-chaining algorithm. Figure 5.1 lists our domain-based lexical-chaining algorithm.

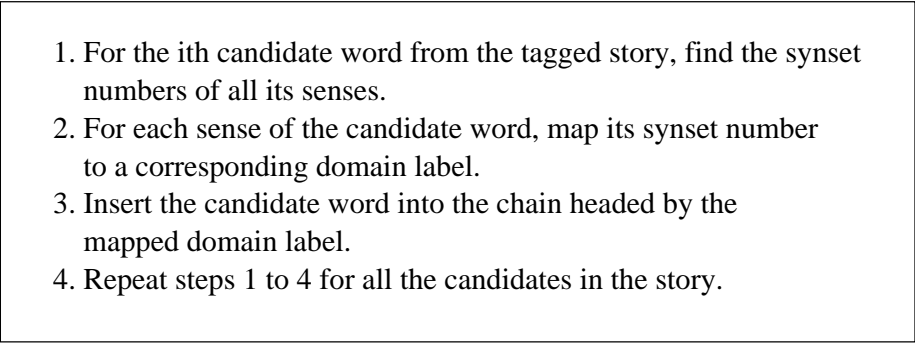
- 
1. For the  $i$ th candidate word from the tagged story, find the synset numbers of all its senses.
  2. For each sense of the candidate word, map its synset number to a corresponding domain label.
  3. Insert the candidate word into the chain headed by the mapped domain label.
  4. Repeat steps 1 to 4 for all the candidates in the story.

Figure 5.1: Domain Chaining Algorithm

The algorithm treats each domain label as the head of its chain, and for each such domain label collects all the candidate words that belong to the same domain. Each such collection forms a lexical chain.

## 5.3 Handling Proper Nouns

In addition to the use of domain labels for lexical chaining, we have also improved the chaining process by introducing a procedure to handle proper nouns. Previous chainers failed to handle these special entities, mainly because they were not defined in the knowledge base used by the chainers.

Recall the TDT definition of a topic. A topic is “a seminal event or activity, along with all directly related events and activities.” The task of topic tracking is to find out whether a story describes events and activities that are directly related to the topic event. Since both events and activities occur at specified time points, in specified location, and with specified people, proper nouns referring to space, time and characters certainly appear in the stories and are valuable for representing the meaning of the stories. In other words, a news story consists of the information on *when*, *where*, *what*, *why*, and *who*. And it is almost always the case that proper nouns contribute to the *when*, *where*, and *who* components of the news story. This shows the value of proper nouns.

One of the issues listed in section 3.7 was the lack of proper nouns from the WordNet database prior to version 1.7. Lexical chainers discussed in both chapter 3 and chapter 4 are based on WordNet, which includes a limited number of proper nouns. Tracking systems based on lexical chains with no proper nouns will not represent the stories precisely. Yet no effective approaches to proper nouns have emerged.

To go about the problem of missing proper nouns in WordNet, Carthy enhances his chainer in Figure 4.1 using term-indexing techniques, so that his tracking system does not leave out the proper nouns in the stories. The rationale behind this move is that “combining multiple indexing techniques is more powerful than relying on just one” [11].

WordNet1.7 added about 20,000 sense keys or senses in its database, a large number of which are proper nouns. However, again we are still facing the same issue: WordNet does not provide for the relationship between the newly added proper nouns.

Our tentative chaining algorithm in Figure 5.1 combines the WordNet database and an extension of it that includes domain labels. But we have left out the proper nouns in the

algorithm, as in most previous chainers. To resolve this issue we propose our revised domain chainer in Figure 5.2 below.

1. For the  $i$ th candidate word from the tagged story, find the synset numbers of all its senses in WordNet.
2. If this candidate word is tagged as a proper noun, insert it into the Proper Noun chain.
3. Else, for each sense of the candidate word, map its synset number to a corresponding Magnini domain label.
4. Insert the candidate word into the chain headed by the mapped domain label.
5. Repeat steps 1 to 4 for all the candidates in the story.

Figure 5.2: Revised Domain-Chaining Algorithm

When the chainer runs, if it finds a proper noun, it goes ahead inserting the proper noun into the chain allocated for proper nouns, without going through the searching procedures in WordNet. In addition to increased performance, this direct insertion also saves a lot of processing time.

#### 5.4 Building Chains for a Sample Story

To illustrate how our domain chainer works, we build chains for the following short passage from CNN [41]. The resulting chains are listed in Table 5.1.

*WASHINGTON (CNN) – Few secretaries of state have been as prominent or controversial as Henry Kissinger, a German-born immigrant who dominated U.S. foreign policy during the Nixon and Ford administrations. Kissinger, 79, was called back into public service Wednesday by President Bush, who appointed him chairman of an independent commission charged with investigating why the United States was not prepared for last year’s terrorist attacks on*

*New York and Washington.*

Our algorithm in Figure 5.2 proceeds by first selecting a candidate word from the tagged story. We use a tagger coupled with a compound identifier to tag the file. The tagged file for the above news story is shown in Figure 5.3.

WASHINGTON NNP
CNN NNP
secretary_of_state NN
Henry NNP
Kissinger NNP
immigrant NN
U.S. NNP
foreign_policy NN
Nixon NNP
Ford NNP
administration NNS
Kissinger NNP
public_service NN
president_bush NNP
chairman NN
commission NN
united_states NNP
year NN
attacks NNS
new_york NNP
Washington NNP
Wednesday NNP

Figure 5.3: Tagged File for the CNN Kissinger Story

Next, for each candidate word, if it is a proper noun, the algorithm inserts it into the chain for the proper nouns. The twenty-two nouns from the Kissinger story and eleven proper nouns are identified. The algorithm collects all thirteen of them in the chain for the proper nouns.

If the candidate word is a noun, the algorithm searches in the WordNet database for all its senses. For each sense, the algorithm connects it with the corresponding domain label and then inserts the candidate into the chain headed by the corresponding domain label. For example, a search in WordNet 1.7 returns the following nine senses for the candidate word *commission*:

CHAIN #	HEAD	CHAIN MEMBER			
3	history	attack	president bush		
30	play	attack			
34	sport	attack			
78	medicine	administration			
120	military	attack			
122	school	year			
142	economy	commission	administration	secretary of state	
145	enterprise	chairman	secretary of state		
150	administration	commission secretary of state	commission secretary of state	administration	administration
151	law	attack	commission	commission	
152	politics	commission administration	commission administration	president bush foreign policy	administration secretary of state
157	factotum	attack commission	attack commission	attack public service	attack
160	time period	year	year	year	
161	person	immigrant			
171	proper nouns	Hudson River Kissinger Kissinger president_bush	Washington Ford Henry	new york Nixon CNN	united states U.S. WASHINGTON

Table 5.1: Chains for the CNN Kissinger Story

1. 06731197 committee, commission – (a special group delegated to consider some matter)
2. 11014235 commission – (a fee for services rendered based on a percentage of an amount received or collected or agreed to be paid ... )
3. 00827476 commission, commissioning – (the act of granting authority to undertake certain functions)
4. 11567112 commission – (the state of being in good working order and ready for operation)
5. 06778977 deputation, commission, delegation, delegacy, mission – (a group of representatives or delegates)
6. 05855743 commission, charge, direction – (a formal statement of a command or injunction to do something)
7. 05318229 commission, military commission – (an official document issued by a government and conferring on the recipient the rank of an officer in the armed forces)
8. 00556349 perpetration, commission, committal – (the act of committing a crime)
9. 00529505 mission, charge, commission – (a task that has been assigned to a person or group)

The algorithm matches synset numbers *00827476* and *06778977* with the domain of *politics*, *06731197* and *05318229* with *administration*, *11014235* with *economy*, *00556349* and *05855743* with *law*, and *00529505* and *11567112* with *factotum*. After processing all the candidate words, the algorithm generates the chains listed in Table 5.1.

## CHAPTER 6

### APPLYING CHAINING TO TRACKING

In this chapter, we discuss in more detail Carthy’s tracking system, which was summarized in section 2.4.1, and present our domain-chaining-based tracking system. Carthy’s tracking system is the first one that utilizes the lexical chaining process. It is also the first system that combines the shallow NLP approach with the statistical term-indexing approach. To test the performance of our system, we implement both systems, with the modifications to Carthy’s original system. Since our system is based on only the shallow NLP approach, we implement the lexical chaining part of Carthy’s original algorithm and replace the term-indexing part of the original algorithm with the proper noun handling methodology explained in section 5.3.

#### 6.1 Tracking with Carthy’s System

Figure 2.3 summarizes the procedures of Carthy’s system. This section explains how Carthy’s system computes the similarity between two news stories by comparing the chain similarities. The *tracking set* in step 1 of Figure 2.3 is the set of all lexical chains built from the  $N_t$  stories that define an event. The *tracking query* in step 2 consists of all lexical chains of the  $N_t$  stories. Let us call each of the  $N_t$  stories a topic story, and each of the new documents a new story. Let us further define the chains computed from the topic story as topic chains, and the chains from the new story as new chains. To compute whether or not a topic story tracks a new story, we need to check if the similarity between the two stories is above a

threshold value. To compute the similarity and the threshold value, the steps below are followed:

1. Compute the similarity between the topic chains and the new chains using the chain similarity function described in Figure 6.1. The function takes two arguments, the topic chains for the topic document and the new chains for the new document. The similarity function needs the Overlap Coefficient formula for chains  $c_1$  and  $c_2$  defined as:

$$OverlapCoefficient = \frac{|c_1 \cap c_2|}{\min(|c_1|, |c_2|)} \quad (6.1)$$

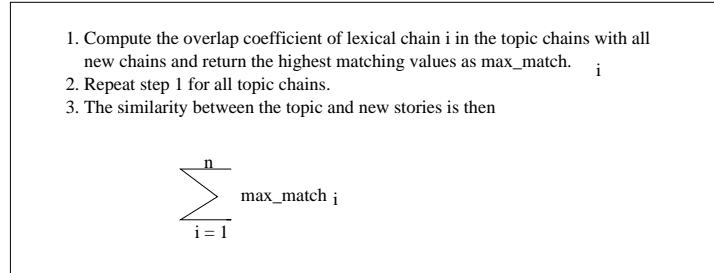


Figure 6.1: Chain Similarity Function

2. Compute the threshold similarity by passing two copies of the chains for the topic story. The similarity function returns the threshold similarity by comparing the topic story to itself.
3. If the chain similarity from step 1 is greater than the product of the threshold similarity and a threshold constant, often set between 0.2 and 0.6, then the topic story tracks the new story.



### 6.1.1 Handling Proper Nouns

Section 6.1 describes Carthy’s tracking system using only the lexical chains. In Carthy’s original design, the term-indexing technique is combined with the lexical chaining approach, to overcome the lack of proper nouns in the WordNet database. Since the major concern of our work is with lexical chaining, we will not combine the statistical approach with the chaining process here. To avoid the issue with proper nouns, we modify Carthy’s chaining algorithm as shown in Figure 6.2. To be consistent with our domain-chaining algorithm shown in Figure 5.2, we replace the stopwords list in step 1 in Figure 4.1 with our tagged file for the stories.

1. For the  $i$ th candidate word from the tagged story, find the synset numbers of all its senses in WordNet.
2. If this candidate word is tagged as a proper noun, insert it into the Proper Noun chain.
3. Otherwise, take the  $i$ th term in the story and using WordNet, generate the set called Neighbor  $i$  of its related synsets using the hyponym/hypernym and meronym/holonym relationships.
4. For each other term in the document, if it is a member of the set Neighbor  $i$  then add its synset identifier to the lexical chain for term  $i$ . The location of the chain element in the document stream is also stored.
5. If the lexical chain contains 3 or more elements then store the chain in a chain index file
6. Repeat steps 1, 2 and 3 for all terms in the story.

Figure 6.2: Modified Carthy Chainer

## 6.2 Implementing Carthy’s Tracking System

We briefly describe the data structures and functions in our implementation of Carthy’s system.

### 6.2.1 Data Structures

The two basic data structures in our implementation are **lex\_Cluster** and **lex\_Chain**.

**lex\_cluster** An array of linked lists to collect all the synonyms, hyponyms, hypernyms meronyms and holonyms for each candidate word from WordNet..

**lex\_Chain** A linked list of linked lists to hold the chains headed by each synset number or sense of all candidate words. One of those chains is dedicated to the proper nouns in the story.

### 6.2.2 Functions

We describe the functions following the logic flow in the tracking system. The system starts with calling the **do\_jc\_tracking** function

#### **do\_jc\_tracking**

1. Fetch a topic story.
2. Call **makeChain** function to build a `topic_lex_chain`.
3. Fetch a new story.
4. Call **makeChain** function to build a `new_lex_chain`.
5. Call **compSimilarity** function to compute the similarity between the `topic_lex_chain` and the `new_lex_chain`.
6. Call **file\_tracking** function to check if the new story is tracked by the topic story with threshold constant set between 0.2 and 0.6.

## **makeChain**

The **makeChain** function works as follows:

1. Call **compute\_tagged\_file** function, which combines Milhalcea's tokenizer and tagger and returns a file where each word from the story is tagged with the relevant part-of-speech tag.
2. Call **startBldChain** function which returns the lexical chains for a story.

## **startBldChain**

The **startBldChain** function works as follows:

1. For each noun candidate from the tagged file, if it is a proper noun, call **addNNPtoChain** to insert it into the chain allocated for the proper nouns.
2. Otherwise, if the senses of this noun have not been inserted into the chains yet, call **bldCluster** to find all the synonyms, hyponyms, hypernyms meronyms and holonyms for this candidate word.
3. Then call **bldChain**, which builds the chains corresponding with each sense of the candidate word using the lexical cluster from previous step.
4. Otherwise, if duplicates exist, insert all senses of this noun candidate into the corresponding existing chain.

**bldCluster** The **bldCluster** function proceeds as follows:

1. Call **addHyper** to search for all hypernyms in WordNet for the candidate noun.
2. Call **addHypo** to search for all hyponyms in WordNet for the candidate noun.

3. Call **addHasPart** to search for all meronyms in WordNet for the candidate noun.
4. Call **addIsPart** to search for all holonyms in WordNet for the candidate noun.

All found items are stored in the `lex_cluster` for the candidate word.

**bldChain** Check if each of the rest of the candidate words appears in the cluster of the current candidate word, If a hit occurs, insert the candidate word into the chain of headed by this synset number of the current candidate word.

**compSimilarity** The function computes the similarity between the topic chain and the new chain using the method in Figure 6.1. However, this method has a complexity of  $O(n^2)$ .

### 6.3 Tracking with Domain Chaining

The generic task of topic tracking consists of text representation, formulating a classifier, and comparing the subsequent stream of stories to the classifier, so that the documents that are similar enough to the classifier are retrieved [32]. Carthy's tracking algorithm, as shown in Figure 2.3 follows the generic task closely. Representing a story as lexical chains corresponds to the task of text representation. That is, the topic or the new story is represented as a set of lexical chains. Steps 1 and 2 correspond to the task of formulating a classifier. That is, the topic chain functions as the classifier for the stream of stories. Steps 3 and 4 correspond to the task of comparison. Due to this closeness of Carthy's system to the generic task, our domain tracking system follows the same steps listed in Figure 2.3.

### 6.3.1 Computing Similarity

Given the similarity mentioned above, our domain tracking system is significantly different from Carthy's, in that in the core of the domain-tracking system is the domain chainer, as shown in Figure 5.2. We also compute similarity between the topic story and the new story differently. To compute if the topic story tracks the new story, we follow these steps:

1. Compute the similarity between the topic story and the new story using the chain similarity function illustrated in Figure 6.3. The function takes two arguments, topic and new chains built from the algorithm in Figure 5.2. This function relies on the Overlap Coefficient formula for chains  $c_1$  and  $c_2$  defined as:

$$OverlapCoefficient = \frac{|c_1 \cap c_2|}{\min(|c_1|, |c_2|)} \quad (6.2)$$

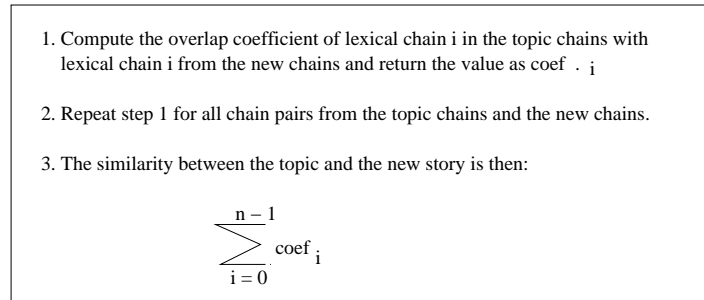


Figure 6.3: Domain Chain Similarity Function

2. Compute the threshold similarity by passing two copies of the chains for the topic story. The similarity function returns the threshold similarity by comparing the topic story to itself.
3. If the chain similarity from step 1 is greater than the product of the threshold similarity

and a threshold constant, often set between 0.2 and 0.6, Then the topic story tracks the new story.

## 6.4 Implementing Domain-Tracking System

We briefly describe the data structures and functions of our domain tracking system in this section.

### 6.4.1 Data Structures

The system uses the following main data structures:

**index\_table** A hash table where each of the domain labels from Magnini’s 167 domains is assigned a unique index value from 0 to 166.

**label\_table** A hash table where each synset number of a word’s senses is mapped with its domain label and its unique index value from the **index\_table** above.

**lex\_chain** An array of linked lists where the chains are collected for all candidate words from a story. The array size is set to a constant of 168, with each index associated with one of the 167 domain labels. The words collected under the index or domain label form a chain. The last index of the **lex\_chain** is reserved for the proper nouns.

### 6.4.2 Functions

We describe the functions in the domain tracking system in the context of system organization. The system starts with calling the **do\_dm\_tracking** function on the top level.

**do\_dm\_tracking**

1. Call **fillIdxTbl** to instantiate the **index\_table**.
2. Call **fillSsMapTbl** to instantiate the **label\_table** using **index\_table** and using the map between WordNet1.7 synset numbers and domain labels.
3. Fetch a topic story.
4. Call **compTagFile** to tokenize and tag the topic story.
5. Call **bldChains** function to build a **topic\_lex\_chain**.
6. Call **cmpThrsh** to compute a threshold similarity by comparing the topic story with itself.
7. Fetch a new story.
8. Call **compTagFile** to tokenize and tag the new story.
9. Call **bldChains** function to build a **new\_lex\_chain**.
10. Call **compSimilarity** function to compute the similarity between the **topic\_lex\_chain** and the **new\_lex\_chain**.
11. Call **file\_tracking** function to check if the new story is tracked by the topic story with threshold constant set between 0.2 and 0.6.

**bldChains** **bldChains** works as follows:

1. If the candidate word is a proper noun from the part-of-speech tagged story, insert it into the last chain of the **lex\_chain**.
2. Otherwise, find all synset numbers corresponding to all senses of the candidate word.

3. Retrieve from the **label\_table** the index value that each synset number mapped with.
4. Insert the synset number and the candidate word into the chain headed by the mapped domain label.

**compSimilarity compSimilarity** computes the similarity between two stories using the function described in Figure 6.3. The upper bound of this function is  $O(n)$ .

## 6.5 Hypotheses

We have presented both Carthy’s tracking system and our domain-chain tracking system in previous sections of this chapter. Since we are mainly concerned with enhancing lexical chaining using WordNet with extra databases in current work, for comparison purposes, we have only implemented the lexical chaining part of Carthy’s original algorithm. The other part of Carthy’s algorithm, term indexing, is designed to compensate for the lack of proper nouns in WordNet. To make up for the absence of term indexing from our implementation we have added our methodology for handling proper nouns to Carthy’s chaining algorithm. Figures 5.2 and 6.2 show that we handle proper nouns in the same way in both chainers. Given the same methodology for handling proper nouns, we expect both tracking systems to be able to resolve the proper noun issue. However, we still expect that our domain tracking system will outperform Carthy’s system in two respects.

First, our tracking system is designed to resolve the issues listed in section 3.7. Our tracking system combines an extra database, the Magnini domains, with the WordNet database.



As in section 4.4, the three features of the Magnini domains provide means to link the databases of different parts of speech, connect the lexical items associated by non-systematic semantic relationships, and shorten the distance between semantically related items that are far apart in the WordNet database. Since the domain chainer, the core of our tracking system, is based on these features of Magnini domains, we expect our tracking system to be able to deal with the issues in section 3.7. On the other hand, Carthy’s chainer relies on only the WordNet database, so it is not much different from previous chainers in terms of handling the semantic relationships between lexical items.

Second, the domain tracking system is much more efficient than Carthy’s system, in that our system, like Silber and McCoy’s algorithm, is a linear system. When we consider the time complexity of the systems, we only count that of our code. We do not have to add in the time complexity of the internal operations of WordNet, Mihalcea’s tokenizer and her compound identifier. Our informal proof goes as follows:

Two factors make Carthy’s system an at least  $O(n^2)$  system. One, the **startBldChain** builds for each of the  $m$  senses of the  $n$  candidate word a cluster by calling the **bldCluster** function, which in turn for each of the  $m$  senses calls the **addHyper**, **addHypo**, **addHasPart**, and **addIsPart**. This chain of operations takes at least  $n * M$  steps and approaches  $O(n^2)$  provided that we only search up or down only one level of the WordNet noun tree hierarchy for the hypernyms, hyponyms, meronyms, and holonyms. In fact, in our implementation of the **bldCluster** function, we searched up and down all levels of the tree hierarchy to collect all the the hypernyms, hyponyms, meronyms, and holonyms of a word’s sense. This will lead to a higher complexity than  $O(n^2)$ . We just go by the lower bound here. Two, the **compSimilarity** function is also a  $O(n^2)$  because this function compares  $n$  set of chains

to another  $n$  set of chains for two news stories, as shown in Figure 6.1.

The domain tracking system is a  $O(n)$  system for several reasons. First, unlike Carthy's system, for each candidate word, our system only needs to retrieve its synsets and maps them with the right domains. Since we do not search up or down the Wordnet hierarchy tree, the retrieving process for  $n$  candidate words takes time  $O(n)$ . Since we use the functions **fillLexTbl** and **fillSsMapTbl**, which are based on hash tables **index\_table** and **label\_table** respectively, the mapping process is also linear. Second, to compute the similarity, function **compSimilarity** compares each pair of the two sets of 167 chains that have the same index numbers. If we assume that each chain contains  $n$  elements, then the worst case is that the similarity computing process takes  $167n$  steps. That is also  $O(n)$ . Therefore, the upper bound of our tracking system is  $O(n)$ .

In chapter 7, we presents the experiments done to compare the performance of Carthy's system and our domain-tracking system, and the results from these experiments.

## CHAPTER 7

### EXPERIMENT AND EVALUATION RESULTS

To compare Carthy’s system and ours, we have carried out four sets of experiments. In the first set of experiments, we checked the results of Carthy’s chainer and our chainer for the sample text quoted in section 3.6.2 above. In the second experiment, we ran Carthy’s system without our proper-noun-handling methodology for the 100 topics from the Reuter’s corpus for TREC 2002. In the third experiment, we ran Carthy’s system with the proper-noun-handling technique for the same 100 topics. We also ran our domain-tracking system for the same 100 topics in the fourth experiment. We treated the systems in the experiments as three separate systems and computed the  $T11SU$ ,  $T11F$ ,  $Precision$ ,  $Recall$ ,  $Pmiss$ ,  $Pfa$ , and  $C_{det}$  scores for each system. Since we have not used the latest TDT corpora for our experiments, we will not try to interpret the  $Pmiss$ ,  $Pfa$ , and  $C_{det}$  scores.

#### 7.1 Comparing Chaining Results

We ran Carthy’s chainer and our chainer on the sample text given in section 3.6.2. The resulting chains are listed in Tables A.1, A.2, A.3 and A.4 of appendix A. There are no proper nouns in this text. In this experiment, we did not pick a text with proper nouns for two reasons: First, since St-Onge had computed the chains with his greedy chainer and provided the results in `citestongethesis`, it would be advantageous for us to compare the results from our chainer with his while using the same text. And St-Onge happened to pick a text without proper nouns for his chainer. Second, our chainer inserts proper nouns directly

into a pre-specified chain without going through the disambiguation process. We attempt to compare this process with Carthy’s chainer and St-Onge’s here. Proper-noun handling is not a key issue for this experiment.

Tables A.1 and A.2 in appendix A list chains with two or more elements that resulted from running Carthy’s chainer shown in Figure 6.2 over the sample text from section 3.6.2. Since Carthy’s chainer is not greedy, it keeps all possible chains headed by each sense of a candidate noun. Due to this, given that we have removed all chains with fewer than 2 elements, we have still kept 40 chains in Tables A.1 and A.2 for the short passage mentioned above, while St-Onge’s greedy algorithm yields nine chains for the same passage. See section 3.6.2 for these chains.

It seems that in this experiment, Carthy’s chainer has overcome the *over chaining* problem. Previously, in St-Onge’s chains obtained from the same sample text, candidate words *velocity* and *simultaneity* are grouped into the same chain as *train*. *Body*, and *figure* are collected into one chain. According to St-Onge, these are cases of *overchaining*. Since Carthy’s chainer considers all the associated senses of a candidate word, *over chaining is avoided*.

However, in Tables A.1 and A.2 we cannot find a chain for candidate word *embankment*, which is supposed to be in the chain headed by one of the senses of *train*. Neither can we find a chain for *velocity*, which is expected to be in the same chain as *traveling 213849*. These are cases of *underchaining*, due either to the missing of a semantic relationship or to a large distance between some lexical items in WordNet database.

Tables A.3 and A.4 in appendix A list the domains which resulted from applying our domain chainer to the same sample text in section 3.6.2. It seems as if the problematic candidate words above have been correctly disambiguated and assembled into the appropriate

chains. We tested Carthy’s chainer and our domain chainer on dozens of news stories from various sources and had similar results. These experiments indicate that the domain chainer resolves the issue of over- and underchaining that resulted from using Magini’s domains as an extension to the WordNet database.

## 7.2 Comparing Tracking Systems

To compare the performance of Carthy’s tracking system and our domain tracking system, we performed the following three sets of experiments:

- We ran Carthy’s system without including proper nouns.
- We ran Carthy’s system including proper nouns.
- We ran the domain-tracking system including proper nouns.

Let’s treat the systems of the three experiments as three systems. Here are the specifications of our experiments:

**Topics** All three sets of experiments took the 100 topics from Reuters Corpus Volume I as input.

**Threshold Constants** The threshold constant was set to 0.2, 0.3, 0.4, 0.6, 0.6, and 0.8, respectively, for each experiment, for all 100 topics.

**Statistics Computing**  $C_{Det}$ ,  $P_{Miss}$ , and  $P_{Fa}$  defined in Table 2.1 were computed for each topic.  $T11SU$ ,  $T11F$ ,  $Precision$ , and  $Recall$  defined in Table 2.2 were computed for each topic. Average values were computed across all 100 topics.

### 7.2.1 $T11SU$ and $T11F$ Scores

Figure 7.1 plots the  $T11SU$  scores for the three experiments listed above with the threshold constant set to 0.4. Figure 7.2 lists the  $T11F$  scores for the same experiments with the threshold constant set to 0.4. Figures B.1, B.3, B.5, and B.7 in appendix B show the  $T11SU$  scores with threshold constant set to 0.2, 0.3, 0.6 and 0.8, respectively. Figures B.2, B.4, B.6, and B.8 in appendix B show the  $T11F$  scores with threshold constant set to 0.2, 0.3, 0.6 and 0.8, respectively.

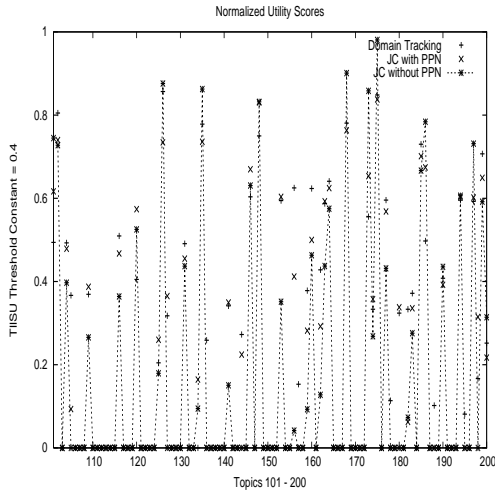


Figure 7.1:  $T11SU$   $tc = 0.4$

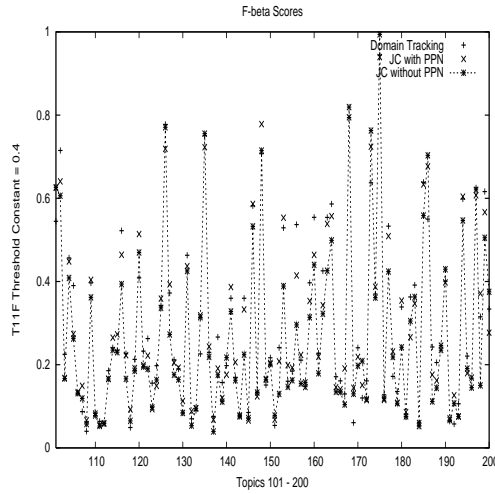


Figure 7.2:  $T11F$   $tc = 0.4$

In Figures 7.1, B.1, B.3, B.5, and B.7, it is obvious that Carthy's tracking system without taking proper nouns into account gets the lowest  $T11SU$  scores, as indicated by the solid line in the figures. It is not easy to tell the difference between the  $T11SU$  scores for other two systems and the  $T11F$  scores for all three systems in the figures above. We will look at the average  $T11SU$  and  $T11F$  scores for all three systems in section 7.2.4.

### 7.2.2 Precision-Recall and DET Plots

Figure 7.3 is the precision-recall plot of the performances of the three systems with the threshold constant set to 0.4. Figure 7.4 is the DET plot of the systems with threshold constant set to 0.4. Figures C.1, and C.3 in appendix C are the precision-recall plot with the threshold constant set to 0.6 and 0.8 respectively. Figures C.2, and C.4 in appendix C are the precision-recall plots with the threshold constant set to 0.6 and 0.8, respectively.

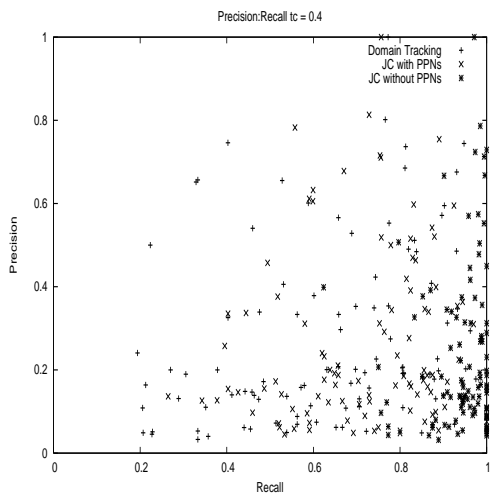


Figure 7.3: Precision-Recall  $tc = 0.4$

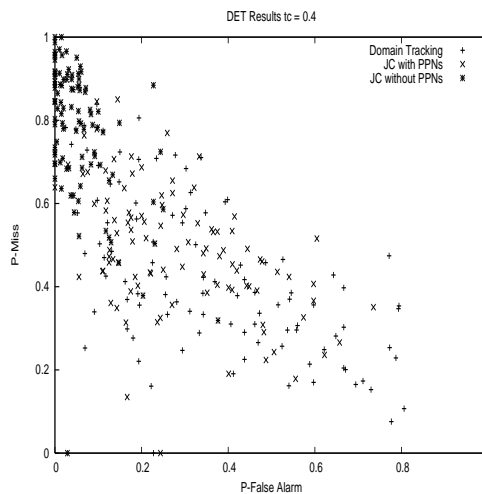


Figure 7.4: DET  $tc = 0.4$

### 7.2.3 Average Scores

Table D.1 in appendix D lists the average scores across all 100 topics for Carthy's tracking system without taking proper nouns into account.

Table D.2 in appendix D lists the average scores across all 100 topics for Carthy's tracking system taking proper nouns into account.

Table D.3 in appendix D lists the average scores across all 100 topics for our domain-tracking system with the proper noun handling methodology, taking proper nouns into account.

The maximum thresh constant is set to 1.0 in all three tables.

#### 7.2.4 Comparing Average T11SU Scores

Using the data from Tables D.3, D.2, and D.1 in appendix D, we plot the average T11SU scores for the three systems in Figure 7.5.

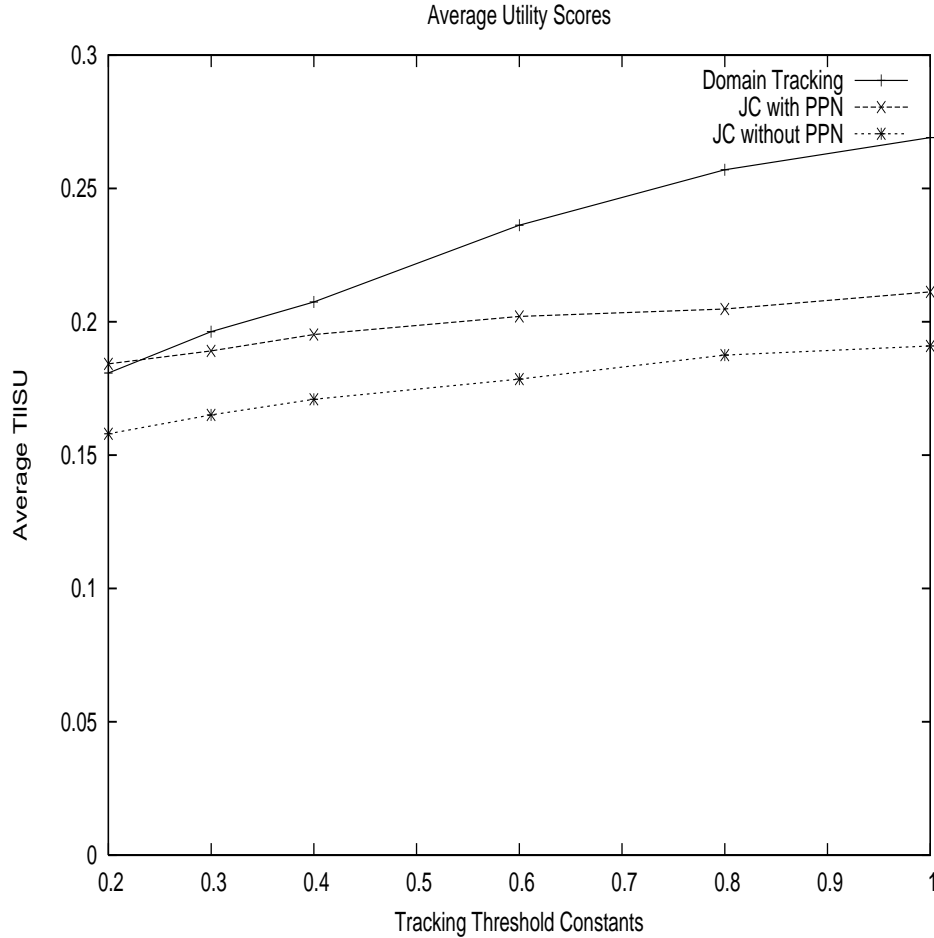


Figure 7.5: Average T11SU Scores

In Figure 7.5, Carthy's system with NNP shows an average 13 percent utility increase



from Carthy’s system without NNP for all threshold constants. The domain-tracking system gains an average of 18.6 percent utility over Carthy’s system with NNP, for all threshold constants. The domain-tracking system shows an average of 28.1 percent utility increase over Carthy’s system with NNP, for all threshold constants.

### 7.2.5 Comparing Average T11F Scores

We plot the the average T11F scores for the three systems in Figure 7.6, using the data from Tables D.3, D.2, and D.1 in appendix D.

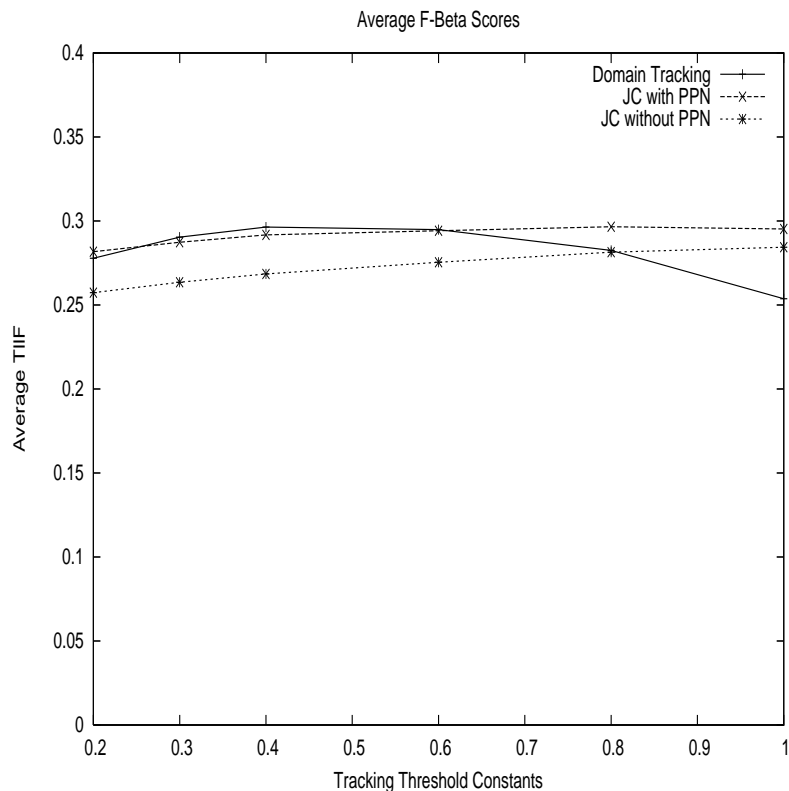


Figure 7.6: Average T11F Scores

Because the domain-tracking system has shown obvious utility increases from Carthy’s system, we expect to see an increase of  $T11F$  scores for the domain-tracking system in Figure 7.6. But in Figure 7.6, although the domain-tracking system starts out with slightly higher

$T11F$  scores than the other two systems, it drops below Carthy’s system with NNP after the threshold constant 0.6, and below Carthy’s system without NNP after 0.8.

tc	Carthy’s without NNP	Carthy’s w/ NNP	Domain Tracking
0.2	33435	24120	26515
0.3	31962	22010	21538
0.4	30539	20150	17724
0.6	27855	17232	12110
0.8	25387	14978	8590
1.0	23264	13200	6252

Table 7.1: Total Files Returned by Each System

To explain the decrease of  $T11F$  scores for the domain-tracking system, we will first examine the total number of files tracked by each system in Table 7.1. In this table, Carthy’s system without NNP returns the most files. The domain-tracking system tracks the fewest. Carthy’s system with NNP scores between these two. Now recall that in formula 2.10, defined in Table 2.2, the denominator is  $0.25 * R - +N + +1.25 * R+$ . For both of Carthy’s systems,  $N+$ , which refers to the number of files that are retrieved but are not relevant, can be much bigger than that for the domain-tracking system. Given this,  $T11F$  for the latter ought not to drop below that for the previous two systems. This is something that will require further study.

## 7.2.6 Precision and Recall Scores

Figure 7.7 shows the average precision scores for the systems. Figure 7.8 shows the recall scores for the systems.

Figure 7.7 shows that Carthy’s system with NNP gives higher precision scores than the one without NNP. It also shows that the domain-tracking system attains higher precision

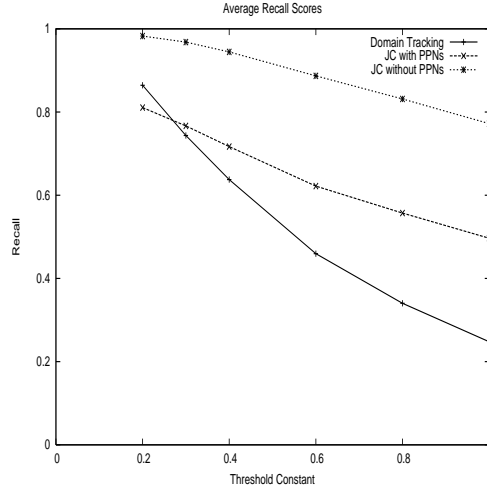
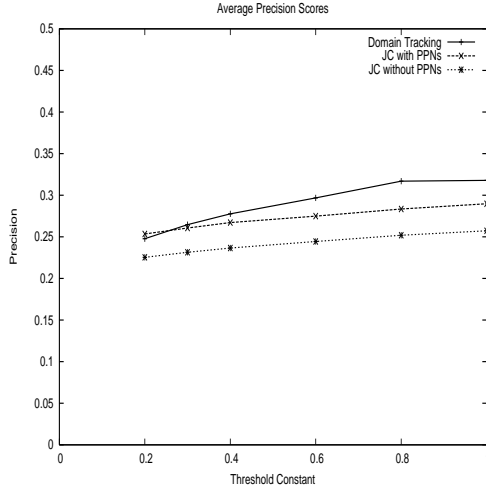


Figure 7.7: Average Precision Scores      Figure 7.8: Average Recall Scores

scores than the other two systems. This indicates a better performance of the domain-tracking system.

In Figure 7.8, the system without NNP has the highest recall scores. The domain-tracking system has the lowest recall scores. Carthy's system with NNP has intermediate scores. This is a situation similar to that in last section; the necessity for future research is indicated here also.

### 7.2.7 Average $P_{miss} - P_{fa}$ Scores

The average  $P_{miss} - P_{fa}$  scores are shown in Figure 7.9. As stated earlier, we will not try to interpret these scores.

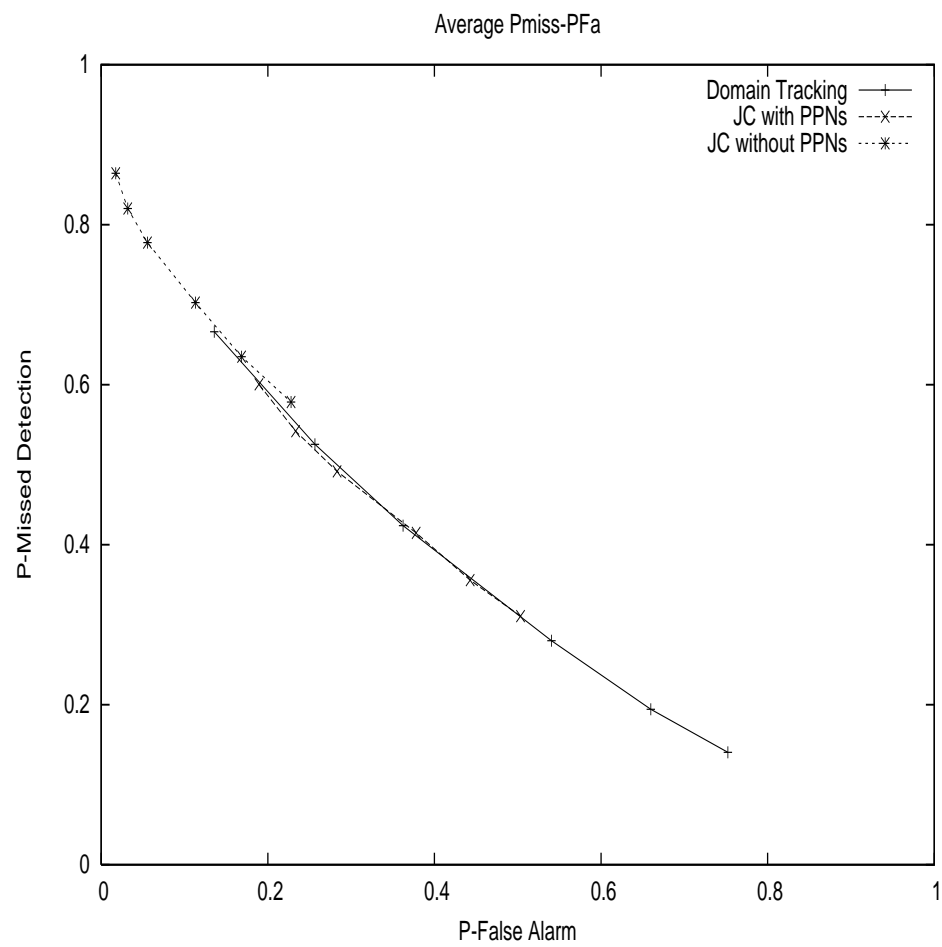


Figure 7.9: Average DET Scores

## CHAPTER 8

### CAN LEXICAL CHAINS REPRESENT TEXTUAL MEANING

As discussed in chapter 3, several information processing systems, such as hypertext linking, text summarizing, text segmentation, information retrieval, and multimedia indexing, have used lexical chainers as the system core, basing on the widely agreed assumption that lexical chains contribute to document texture and hence represent the meaning of documents. This assumption originated in Morris and Hirst's elaboration on the role lexical chains play in forming textual meaning and on the computational feasibility of lexical chains in automatic information systems [31]. Morris and Hirst, in return, ground their elaboration on Halliday and Hasan's textual cohesion in [18]. It is true that Halliday and Hasan's work in the late 1970's precisely detailed the structures contributing to cohesion in English texts and made the assumption that this cohesion formed the texture and thus the meaning of the text. However, it is still not safe to state that lexical chains, as one of the structures that produce cohesion, completely represent textual meaning. In this chapter, we examine three areas which indicate why lexical chains themselves cannot represent the meaning and topics of documents:

- Halliday and Hasan's work on nonstructural textual resources
- static and dynamic cohesion as signals of meaning/texture development
- Martin's discourse semantic structure

## 8.1 Cohesion Helps to Create Text

In this section, we first recapitulate Halliday and Hasan's inventory of cohesive resources. We then explain their statement on cohesion being a necessary condition for textual meaning. We also show that their views on cohesion and textual meaning have not been proved in their book. Following that, we show how Morris and Hirst base their work on Halliday and Hasan's. Finally, we discuss the use of lexical chains in our and other information processing systems..

### 8.1.1 Cohesive Resources

According to Halliday and Has-an, cohesion refers to relations of meaning that exist within the text, and that define it as a text.

In [18], the inventory of resources that contribute to relations of meaning is organized as:

- reference
- ellipsis
- substitution
- conjunction
- lexical cohesion

**Reference** refers to resources for referring to a participant or circumstantial element whose identity is recoverable. In English the relevant resources include demonstratives, the definite article, pronouns, comparatives, and the phoric adverbs *here*, *there*, *now*, and

*then*. **Ellipsis** refers to resources for omitting a clause, or some part of a clause or group, in contexts where it can be assumed. In English conversation, rejoinders are often made dependent through omissions of this presupposable information *Did they win? - Yes, they did*. Resources of place holders are referred to as **substitution**, e.g. *so* and *not* for clauses, *do* for verbal groups, and *one* for nominal groups. A much larger inventory of connectors than previous three resources which link clauses in discourse are referred to as **conjunction**. This resource comprises linkers which connect sentences to each other, but excludes paratactic and hypotactic (coordinating and subordinating) linkers within sentences [18, 35].

Halliday and Hasan distinguish between grammatical cohesion and lexical cohesion. The foregoing four inventory items form the techniques of grammatical cohesion. The complement of grammatical cohesion involves open system items, and so is referred to as **lexical cohesion**. Here the repetition of lexical items, synonymy or near-synonymy (including hyponymy), and collocation are included.

### 8.1.2 Cohesion as a Necessary Condition for Meaning

The resource inventory discussed in the above section provides for cohesion in the concept of text which in return helps to create text. Therefore, as pointed out by Halliday and Hasan [18], cohesion is a necessary condition for the creation of text. Cohesion forms one part of the textual, or text-forming, component of the linguistic system that creates text. The textual component creates text, as opposed to non-text, and therein lies its meaning. Within the textual component, cohesion plays a special role in the creation of text. Cohesion expresses the continuity that exists between one part of the text and another. We quote below from Halliday and Hasan [18] in order to finish our presentation on how cohesion contributes to

textual meaning.

*The continuity that is provided by cohesion consists, in the most general terms, in expressing at each stage in the discourse the points of contact with what has gone before. The significance of this lies in the simple fact that there are such points of contact: that some entity or some circumstance, some relevant feature or some thread of argument persists from one moment to another in the semantic process, as the meanings unfold.*

Halliday and Hasan's view points on the relationship between cohesion and meaning is appealing. However, they did not present their points till the last chapter of their book. Then they did not have much room in the book to show how exactly the inventory contributes to textual meaning. It is Martin in [25] in 1992 who gave this a formal justification. We discuss Martin's work in section 8.3.1.

### 8.1.3 Lexical Chains and Textual Meaning

Halliday and Hasan's theories on cohesion and meaning provide a background to Morris and Hirst's model for computing textual meaning using lexical chains. Morris and Hirst's model in return inspired the building of lexical-chain-based information systems mentioned at the beginning of this chapter and our topic-tracking system. All these systems made the same assumption: lexical chains represent the meaning of documents. However, designers of these systems seem to have neglected two points. First, lexical chains, extracted using lexical cohesion, are from only one of the two cohesion systems, the other being grammatical cohesion as discussed in section 8.1.1. Thus, lexical chains represent textual meaning partially. If information is solely based on such lexical chains, then we cannot expect good performance in such systems. Second, Halliday and Hasan did not formally prove how cohesion techniques



contribute to meaning. Their description of the relationship between cohesion and meaning remains an assumption. If a system is based on such an assumption, then the system is not linguistically reliable.

## 8.2 Static and Dynamic Ties

By mid-80's, many linguists and composition theorists reached the conclusion that if good writing implies coherent meaning, and if coherence is expressed partially through linguistic cohesion, it is useful to analyze cohesion in writing as it contributes to coherent meaning in prose. This conclusion is in agreement with Halliday and Hasssan. However, more importantly, a few of these linguists and composition theorists attempted to analyze systematically how cohesion contributes to meaning. Among them is Hartnett.

The relationship between a cohesive item and the item it presupposed in a text is referred to as a cohesive tie [35]. According to Hartnett in [13], cohesive ties vary in the kinds of mental processes they can express; many ties simply hold a reader's attention on a topic, while others develop a topic rhetorically. These differences suggest a reorganization of the linguistics subclasses of Halliday and Hasan's taxonomy of cohesive devices in order to distinguish mental processes and to define more explicitly the effects of cohesive devices on coherence. To achieve this aim, Hartnett proposes two subclasses of cohesive links, **static** and **dynamic ties**, which work together to focus on a topic and develop it.

Static ties hold attention on a topic without necessarily manipulating it or changing it in any way. Many kinds of static ties are essential for achieving unity in English text. They are pervasive regardless of topic or mode of development. They reflect control of the resources for textual structure in text. In [13], Hartnett lists following techniques for static ties:

- using repetition of the same lexical item, demonstratives, third-person pronouns, definite articles, and nominal, verbal, and clausal substitution and ellipsis
- continuative conjunctions, e.g. *well*, and additive conjunctions, e.g. *also*
- parallel structures, which depend on the selection of appropriate grammatical forms as well as on semantic balance
- choice of tense

Dynamic ties are used to express rhetorical manipulation of the topic. They specify how a writer manipulates an internal representation of environment. Instead of repeating an idea, they indicate how it develops, changes, or relates to something else: to a higher class or a specific example, to a cause or a consequence, or to a similar concept with some difference, such as the same item occurring at a different time. Dynamic ties can be formed by many kinds of text features such as temporal conjunctions, hyponymy, causal conjunctions, adversative conjunctions (e.g. *but*, *however*), and the comparative and superlative forms of adjectives and adverbs.

The actual technique inventory lists for both kinds of ties are much longer than what we have listed above. The point to make here is that assuming Hartnett arguments are sound, the lexical chainers built for the information systems can hardly catch the meaning of text given the complexity that we learn of from Hartnett's theories in building a topic and developing it..

We present Hartnett's work here to show the effort made to relate cohesive devices to meaning. In the following, we will see Martin's formal discussion on the relationship between linguistics features and textual meaning.

### 8.3 Discourse Semantic Structure

Martin’s work [35, 25] concentrates on formally describing the semantics of the cohesive resources (devices in Hartnett’s terminology) and their relation to textual meaning. Martin reformulates cohesion as a set of discourse semantic systems at a more abstract level than lexicogrammar. He thus reworks Halliday and Hasan’s nonstructural textual resources as semantic systems concerned with discourse structure, comprising:

- identification
- negotiation
- conjunction
- ideation

In the following, we first define these semantic systems as in [35, 25]. Then we describe how each system corresponds to some type of textual meaning. We then describe a textual meaning model based on these systems. Finally, we show why lexical chains per se cannot represent textual meaning in Martin’s systems.

#### 8.3.1 Systems of Discourse Structure

Discourse semantics studies text-size semantics rather than clause-size semantics. In order to understand text-size meaning or textual meaning, we would have to make sure we understand the four subsystems listed above.

Identification is concerned with resources for tracking participants in discourse. This system subsumes earlier work on referential cohesion in a framework which considers the

ways in which participants are both introduced into a text and kept track of once introduced. In addition, the ways in which phoric items depend on preceding or succeeding co-text, on assumed understandings, or on other relevant phenomena (images, activity, materiality, etc.) are considered.

Negotiation is concerned with resources for exchange of information and of goods and services in dialog. This system subsumes some of the earlier work on ellipsis and substitution in a framework which considers the ways in which interlocutors initiate and respond in adjacency pairs.

Conjunction is concerned with resources for connecting messages, via addition, comparison, temporality, and causality. This system subsumes earlier work on linking between clauses in a framework which considers the ways in which connections can be realized inside a clause through verbs, prepositions, and nouns (e.g. *result in*, *because of*, *reason*). This system also proposes a framework for analyzing internal (pragmatic/rhetorical) and external (semantic/propositional) conjunctive relations, including the possibility of connections realized simply by the contiguity of messages.

Ideation is concerned with the semantics of lexical relations as they are deployed to construe institutional activity. This system subsumes earlier work on lexical cohesion in a framework which considers the ways in which activity sequences and taxonomic relations (of classification and composition) organize the field of discourse. This system proposes a more detailed account of lexical relations - including repetition, synonymy, hyponymy, and meronymy; in addition, collocation was factored out into various kinds of “nuclear” relation, involving elaboration, extension, and enhancement.

### 8.3.2 Textual Meaning Model

With the above systems, Martin formulates a stratum of text-oriented resources dedicated to the analysis of cohesive relations as discourse structure. Once stratified with respect to lexicogrammar, resources resulting from the four systems represent four types of textual meaning:

- textual meaning that corresponds with identification analyses
- interpersonal meaning that corresponds with negotiation analyses
- logical meaning that corresponds with conjunction analyses
- experiential meaning that corresponds with ideation analyses

With the above semantic classification, the study of textual meaning requires a study of patterns of interaction among semantics, lexicogrammar, and phonology/graphology in realization. As far as this interaction is concerned, Martin lists two types of research that concentrate on textual meaning. The first type is cohesive harmony (experiential grammar). The second type is textual grammar (method of development).

Cohesive harmony analysis corresponds to ideation and textual analyses. Cohesive harmony analysis considers how strings and chains interact. The interaction is defined as taking place when two or more members of a string or chain relate in the same way to two or more members of another string or chain. However, as Martin pointed out [35, 25], cohesive harmony analysis is incomplete in various respects as an analysis of textual meaning. For one thing it does not draw on conjunction analysis. Nor does cohesive harmony analysis consider

negotiation. So while it has been proven a remarkably sensitive technique for measuring textual meaning, cohesive harmony analysis is not a sufficient analysis of textual meaning, since in performing such analyses so many relevant parameters of texture can be omitted. Martin completes the proof of Halliday and Hasan's assumption that lexical cohesion is a necessary tool but not a sufficient tool for textual meaning analysis.

To make a complete analysis of textual meaning, the analysis of method of development has to be considered. In this direction, linguists are concerned with the interaction of identification and ideation with information flow in clause grammar. According to Martin, cohesion (cohesive harmony) is simply one aspect of texture, which has to be understood with respect to the interaction of identification, negotiation, conjunction, and ideation with each other and with the lexicogramatical and phonological systems through which they are realized [35, 25].

### 8.3.3 Positioning Lexical Chains

Existing lexical chainers in information systems, including our chainers, perform ideation and identification analyses at their best. However, negotiation and conjunction analyses are missing from these systems. Moreover, the systems have not made any effort to incorporate phonological systems either. Therefore, based on the theories of discourse semantics, once again, existing lexical chainers cannot represent textual meaning completely.

## CHAPTER 9

### CONTRIBUTIONS, LIMITATIONS AND FUTURE WORK

#### 9.1 Contributions

##### 9.1.1 The Chaining Algorithm

Our contributions to lexical chaining have two parts:

- Our domain-chaining algorithm uses the domain labels computed by Magnini as a threading method to collect the lexical items in the same domain under one chain. With this algorithm we have avoided using the fine-grained senses in the WordNet database and have achieved better chaining results than traditional chaining algorithms.
- Our proper-noun handling technique, which is to find all proper nouns in a story and assemble them all into one chain without searching through the WordNet database, is not only effective but also time-saving. This technique has not been employed in previous chainers.

##### 9.1.2 Tracking

Our tracking system is a linear system.

#### 9.2 Limitations

In our experiments, we encountered two problematic issues involved with the domain labels and our tracking system.

### 9.2.1 Semantic Domains

First, Magnini’s domain labels are written as an extension to WordNet 1.6. We used WordNet 1.7 in our work. To use Magnini’s domains, we mapped them to WordNet 1.7 synset numbers. Since WordNet 1.7 has added about 20,000 more senses beyond those in WordNet 1.6, there are about the same number of senses in WordNet 1.7 that have not been provided with any domain labels. At this point, we have not been able to quantify how the absence of these 20,000 or so labels affects our tracking system.

Second, while Magnini’s labels successfully find appropriate domains for most lexical items, there are still some items that are semantically related but are not grouped together using Magnini’s database. One such case is with the lexical items *post office* and *stamp* in the postoffice scenario in section 3.1. Again, we have not been able to quantify how this affects our tracking system.

The two issues above might account for the questions in sections 7.2.1 and 7.2.2.

### 9.2.2 Tracking

Current TDT research requires tracking systems to be able to handle multi-lingual news stories. Our system relies on language-specific semantic databases for extracting semantic relationships between lexical items. Currently, our system is based upon WordNet and its extension for English. If our system is to track stories in Mandarin Chinese or Romanian, then we will have to attach a WordNet database for Mandarin or Romanian. This is possible only if a WordNet database for Mandarin or Romanian is available.



### 9.3 Future Work

Obviously, our future work will aim to overcome the limitations discussed in previous sections.

#### 9.3.1 Semantic Domains

To make best use of semantic domains, we need to extend the semantic domain database so that all synsets in current version of WordNet have corresponding domains.

#### 9.3.2 Combining Statistical and Lexical Chaining Approaches

Carthy combines lexical chaining and term-indexing in his original tracking system. In our future work, we would also like to combine some statistical algorithm with our domain-tracking system.

#### 9.3.3 Evaluations

We plan to replicate the experiments on the TDT corpus.

# APPENDIX A

## CHAINS FROM THE SAMPLE TEXT

CHAIN#	HEAD	CHAIN MEMBER			
1	respect_to 11397895	reference_to 11397895			
2	way 6989847	line 6917073	direction 6989847		
3	way 226596	traveling 213849			
4	point 11427965	direction 11425474			
5	point 11240551	advantage 11240797			
6	line 3131218	way 3899581			
7	line 3368273	way 3899581			
8	line 6918079	point 6941394			
9	place 5268147	place 5268147			
10	place 5322114	place 5322114	point 5321097		
11	place 6941128	place 6941128			
12	place 11521597	place 11521597			
13	place 5275944	place 5275944			
14	place 6963138	place 6963138			
15	place 11521375	place 11521375			
16	place 6942855	place 6942855	point 6941394		
17	place 418179	place 418179	line 414596		
18	place 6887293	place 6887293	point 6941394		
19	place 11505260	place 11505260			
20	place 520733	place 520733			
21	place 6958614	place 6958614			
22	place 4737540	place 4737540			
23	place 6852482	place 6852482			
24	place 6976689	place 6976689	point 6941394		
25	reference_to 11397895	respect_to 11397895			
26	event 9158957	event 9158957			

Table A.1: sample chains 1

CHAIN #	HEAD	CHAIN MEMBER				
27	event 9195879	event 9195879				
28	event 11518383	event 11518383				
29	event 20866	event 20866				
30	advantage 11240797	point 11240551				
31	direction 6989847	way 6989847	line 6917073			
32	rail 3815472	rails 3815472				
33	rail 3456835	line 3131396				
34	traveling 213849	traveling 213849				
35	train 2928030	train 2928030	train 2928030	train 2928030	train 2928030	train 2928030
36	train 3818867	train 3818867	train 3818867	train 3818867	train 3818867	train 3818867
37	train 5946574	train 5946574 event 20866	train 5946574	train 5946574	train 5946574	train 5946574
38	train 6800479	train 6800479	train 6800479	train 6800479	train 6800479	train 6800479
39	train 6815992	train 6815992	train 6815992	train 6815992	train 6815992	train 6815992
40	train 3818471	train 3818471	train 3818471	train 3818471	train 3818471	train 3818471

Table A.2: sample chains 2

CHAIN #	HEAD	CHAIN MEMBER					
69	town_planning	train train	train train	place	place	train	train
70	building_industry	embankment	rail				
74	mechanics	train	train	train	train	train	train
95	anatomy	line	body	body	body	body	
98	chemistry	body					
100	geology	point	point				
108	physics	line	event	event	velocity		
111	electricity	point	point	line			
120	military	point	point	line			
124	publishing	line	reference	reference			
131	telephony	line	line				
135	industry	train train	train	line	train	train	train
137	transport	embankment train body direction train	way train train velocity	way line train rail	way line train rail	train train train traveling	train train traveling train
142	economy	line	reference	use	direction	rail	

Table A.3: sample chains

CHAIN #	HEAD	CHAIN MEMBER					
151	law	use	direction				
152	politics	direction					
154	tourism	way traveling	train traveling	train train	train	train	train
155	fashion	train	train	train	train	train	train
157	factotum	respect_to way point point place place place line place place event event reference train direction	way train point point place place line line place place event event reference use direction	way train point place place line line place place event event reference use direction	way definition point place place line line place place train body reference use rail	way train point place place line line place place train body reference train train	way train point place place line line place place reference_to reference train train train
158	number	point					
160	time_period	simultaneity	point				
162	quality	way point direction	way point	way line	definition use	point use	point advantage
163	metrology	point	line				

Table A.4: sample chains

## APPENDIX B

### MORE T11SU AND T11F FIGURES

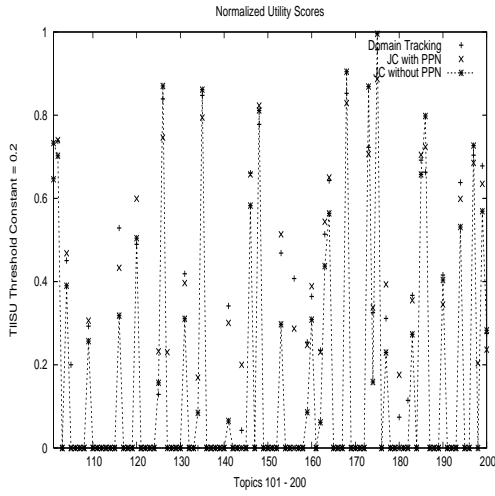


Figure B.1: TlISU  $tc = 0.2$

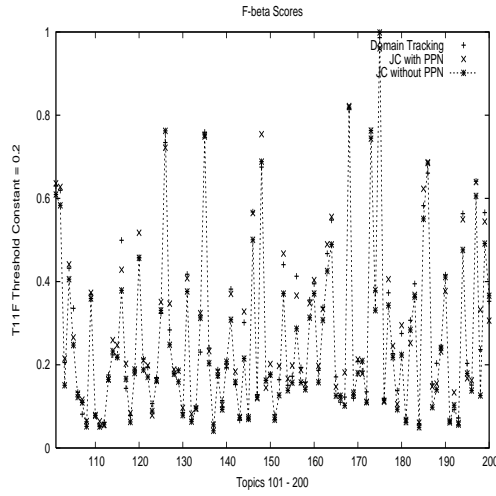


Figure B.2: TlIF  $tc = 0.2$

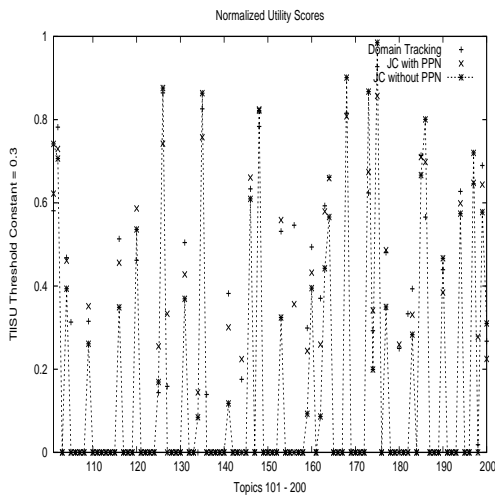


Figure B.3: TlISU  $tc = 0.3$

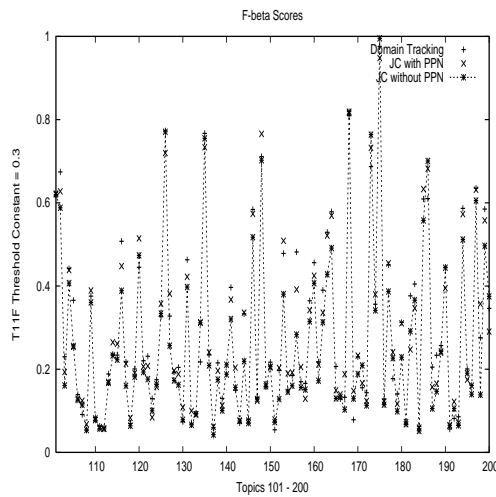


Figure B.4: TlIF  $tc = 0.3$

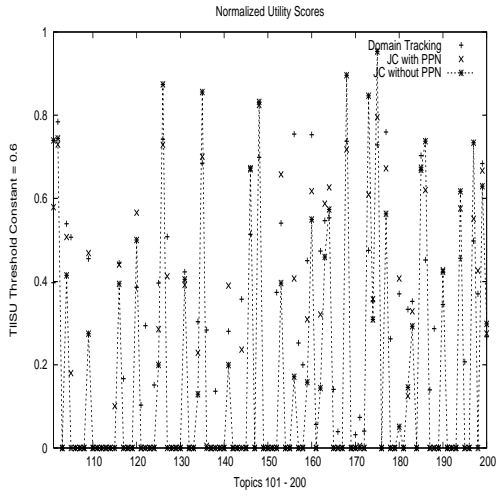


Figure B.5: TlISU  $tc = 0.6$

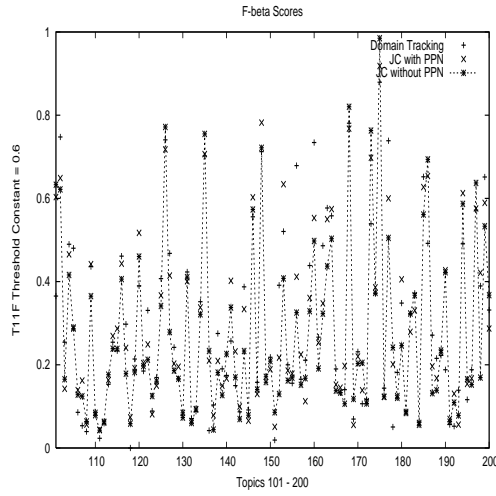


Figure B.6: TlIF  $tc = 0.6$

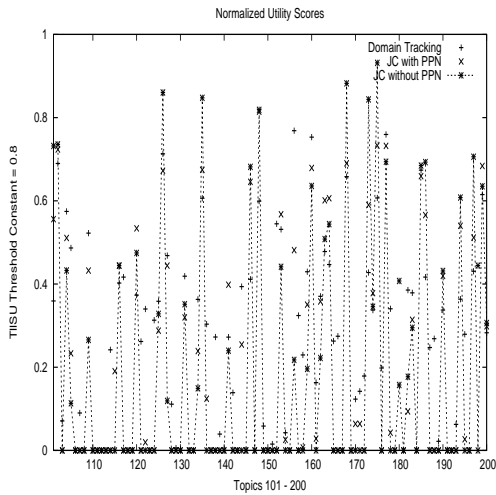


Figure B.7: TlISU  $tc = 0.8$

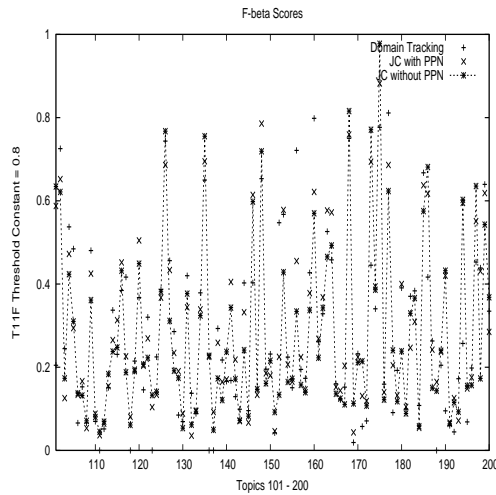


Figure B.8: TlIF  $tc = 0.8$

## APPENDIX C

### MORE PRECISION-RECALL AND DET PLOTS



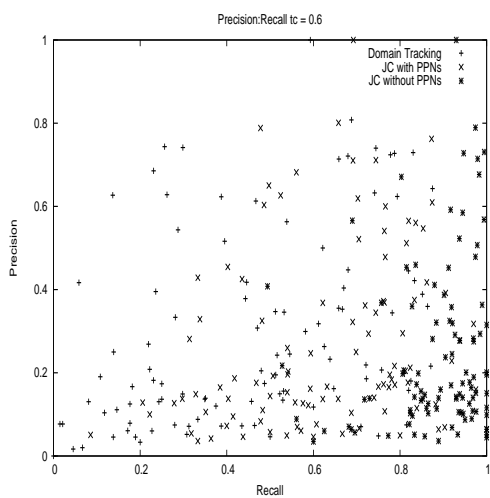


Figure C.1: Precision-Recall  $tc = 0.6$

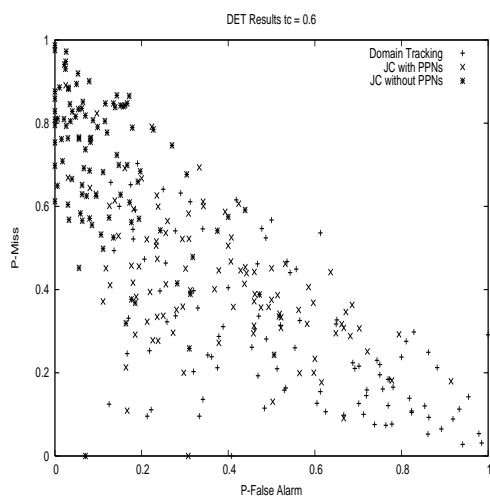


Figure C.2: DET  $tc = 0.6$

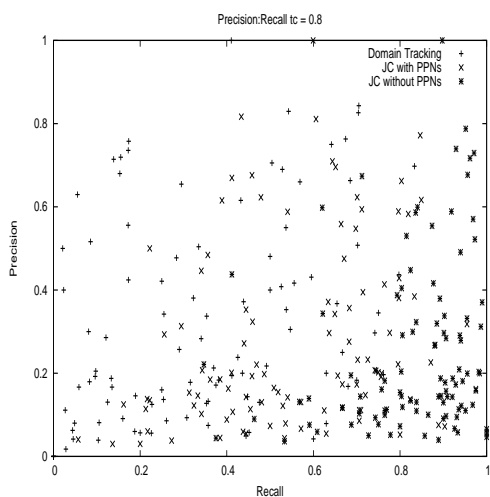


Figure C.3: Precision-Recall  $tc = 0.8$

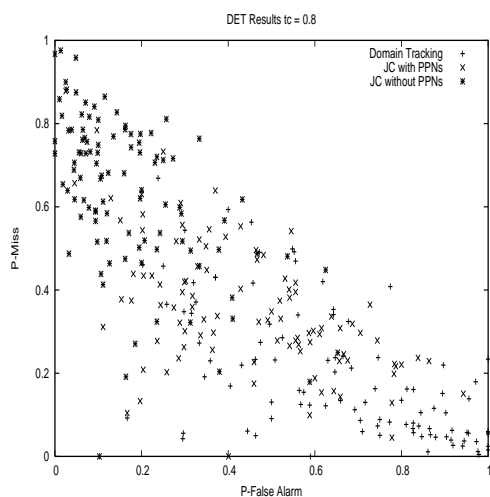


Figure C.4: DET  $tc = 0.8$

## APPENDIX D

### SYSTEM AVERAGE SCORES

Thresh Const	<i>TlSU</i>	<i>TlF</i>	<i>Precision</i>	<i>Recall</i>	<i>Pmiss</i>	<i>Pfa</i>	<i>C<sub>det</sub></i>
0.2	0.1580	0.2573	0.2254	0.9825	0.0175	0.8643	0.7184
0.3	0.1651	0.2635	0.2314	0.9682	0.0318	0.8202	0.7100
0.4	0.1709	0.2685	0.2366	0.9445	0.0555	0.7777	0.7155
0.6	0.1785	0.2754	0.2444	0.8868	0.1132	0.7026	0.7512
0.8	0.1875	0.2814	0.2519	0.8314	0.1686	0.6349	0.7931
1.0	0.1909	0.2844	0.2573	0.7721	0.2279	0.5783	0.8586

Table D.1: JC Average Scores without PPNs

Thresh Const	<i>TlSU</i>	<i>TlF</i>	<i>Precision</i>	<i>Recall</i>	<i>Pmiss</i>	<i>Pfa</i>	<i>C<sub>det</sub></i>
0.2	0.1842	0.2817	0.2534	0.8107	0.1893	0.6005	0.8690
0.3	0.1891	0.2873	0.2606	0.7666	0.2334	0.5418	0.9167
0.4	0.1952	0.2917	0.2671	0.7168	0.2832	0.4916	0.9628
0.6	0.2020	0.2942	0.2749	0.6220	0.3780	0.4148	1.0602
0.8	0.2048	0.2966	0.2835	0.5571	0.4429	0.3556	1.1510
1.0	0.2112	0.2952	0.2898	0.4966	0.5034	0.3107	1.2387

Table D.2: JC Average Scores with PPNs

Thresh Const	<i>TlSU</i>	<i>TlF</i>	<i>Precision</i>	<i>Recall</i>	<i>Pmiss</i>	<i>Pfa</i>	<i>C<sub>det</sub></i>
0.2	0.1808	0.2778	0.2477	0.8643	0.1357	0.6661	0.7980
0.3	0.1963	0.2904	0.2647	0.7435	0.2565	0.5255	0.8972
0.4	0.2074	0.2964	0.2776	0.6376	0.3624	0.4239	1.0171
0.6	0.2362	0.2949	0.2968	0.4595	0.5405	0.2801	1.2440
0.8	0.2570	0.2825	0.3169	0.3402	0.6598	0.1942	1.4361
1.0	0.2691	0.2537	0.3179	0.2478	0.7522	0.1404	1.5956

Table D.3: Domain Average Scores

## BIBLIOGRAPHY

- [1] J. Allan, editor. *Topic Detection and Tracking, Event-based Information Organization*, chapter 1. Kluwer Academic Publishers, 2002.
- [2] J. Allan, editor. *Topic Detection and Tracking, Event-based Information Organization*, chapter 3. Kluwer Academic Publishers, 2002.
- [3] J. Allan, editor. *Topic Detection and Tracking, Event-based Information Organization*, chapter 2. Kluwer Academic Publishers, 2002.
- [4] J. Allan, editor. *Topic Detection and Tracking, Event-based Information Organization*, pages 66–264. Kluwer Academic Publishers, 2002.
- [5] R. Barzilay. Lexical chains for summarization. Master’s thesis, Ben-Gurion University of the Negev, 1997.
- [6] R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *Proceedings of the ACL-97/EACL97 Workshop on Intelligent Scalable Text Summarization*, pages 1017–30, 1997.
- [7] E. Brill. A simple rule-based part-of-speech tagger. In *Proc ACL Conference on Applied Natural Language Processing*, pages 152—155, 1992.
- [8] A. Budanitsky. *Lexical Semantic Relatedness and its Application in Natural Language Processing*. PhD thesis, University of Toronto, 1999.
- [9] A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures, 2001.

- [10] J. Carbonell, Y. Yang, J. Lafferty, R. Brown, T. Pierce, and X. Liu. Cmu report on tdt2: Segmentation detection and tracking. In *Proceedings of the DARPA Broadcast News Workshop*, pages 117–120, 1999.
- [11] J. Carthy and A.F.S. Smeaton. The design of a topic tracking system, 2000.
- [12] J. Carthy and N. Stokes. Lexical chains for topic detection and tracking, 2001.
- [13] B. Coutour, editor. *Functional Approaches to Writing Research Perspectives*, chapter 5. Ablex Publishing Corporation, 1986.
- [14] C. Faloutsos and D. W. Oard. A survey of information retrieval and filtering methods. Technical report, University of Maryland, 1995.
- [15] S. Green. Using lexical chains to build hypertext links in newspaper articles, 1996.
- [16] S. Green. Building hypertext links in newspaper articles using semantic similarity, 1997.
- [17] S. Green. Lexical semantics and automatic hypertext construction, 1999.
- [18] M. Halliday and R. Hasan. *Cohesion in English*. Longman, 1976.
- [19] P. Hatch, N. Stokes, and J. Carthy. Topic detection, a new application for lexical chaining. In *the Proceedings of BCS- IRSG 2000, the 22nd Annual Colloquim on Information Retrieval Research, Cambridge*, pages 94–103, 2000.
- [20] G. Hirst and A. Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion, 2001. URL: <http://www.cs.toronto.edu/~compling/Publications/a-g.html>.
- [21] TDT Homepage. URL: <http://www.nist.gov/speech/tests/tdt/index.htm>.

- [22] R. Kazman, R. Al-Halimi, and M. Mantei. Four paradigms for indexing video conferences. In *IEEE mULTImEDIA*, 3(1), Spring, 1996.
- [23] B. Magnini and C. Strapparava. Experiments in word domain disambiguation for parallel texts. In *SIGLEX Workshop on Word Senses and Multi-linguality, Hongkong*, 2000.
- [24] B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. Using domain information for word sense disambiguation. In *Proceedings of the 2nd International Workshop on Evaluating Word Sense Disambiguation Systems, Toulouse, France, ACL-SIGLEX*, 2001.
- [25] J. R. Martin. *English Text: System and Structure*. Benjamins, 1992.
- [26] R. Mihalcea. *Turning Implicit Knowledge into Explicit Knowledge via Word Semantics: a Model for Information Retrieval*. PhD thesis, Southern Methodist University, 2001.
- [27] R. Mihalcea and D. Moldovan. An iterative approach to word sense disambiguation. In *Proceedings of Flairs 2000, Orlando, FL*, pages 219–223, 2000.
- [28] R. Mihalcea and D. Moldovan. extended wordnet: Progress report, 2001.
- [29] G. Miller. Nouns in wordnet: A lexical inheritance system. *WordNet, An Electronic Lexical Database*, MIT Press, 1998.
- [30] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Introduction to wordnet: An on-line lexical database. *WordNet, An Electronic Lexical Database*, MIT Press, 1998.
- [31] J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of the text. *Computational Linguistics*, 17(1):21–45, March 1991.

- [32] R. Papka. *On-line New Event Detection, Clustering, and Tracking*. PhD thesis, University of Massachusetts, 1999.
- [33] D. Radev and URL: [www.cs.columbia.edu/~radev/cs6998/class/cs6998-11-02/fkattan.ppt](http://www.cs.columbia.edu/~radev/cs6998/class/cs6998-11-02/fkattan.ppt).
- [34] S. Robertson and J. Callan. Guidelines for the trec 2002 filtering track.
- [35] D. Schiffrin, D. Tannen and H. Hamilton, editors. *The Handbook of Discourse Analysis*, chapter 7. Blackwell Publishers, 2001.
- [36] H. G. Silber and K. McCoy. An efficient text summarizer using lexical chains. In *Proceedings of the First International Conference on Natural Language Generation, INLG '2000*, pages 268–271, 2000.
- [37] D. St-Onge. Detecting and correcting malapropisms with lexical chains, msc thesis. Master’s thesis, University of Toronto, 1995.
- [38] M. Stairmand. *A Computational Analysis of Lexical Cohesion with Application in Information Retrieval*. PhD thesis, UMIST, 1996.
- [39] N. Stokes, J. Carthy, and Smeaton A.F. Segmenting broadcast news streams using lexical chains, 2002.
- [40] Overview. URL: <http://trec.nist.gov/overview.html> TREC Homepage.
- [41] 2002 [www.cnn.com](http://www.cnn.com) Wednesday, November 27.