Automatic Summarization of Short Fiction

Anna Kazantseva

Thesis
submitted to the Faculty of Graduate and Postgraduate Studies
in partial fulfillment of the requirements for the degree of
Master of Computer Science

December 2006

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

Abstract

This work is an inquiry into automatic summarization of short of fiction. In this dissertation, I present a system that composes summaries of literary short stories employing two types of information: information about entities central to a story and information about the grammatical aspect of clauses. The summaries are tailored to a specific purpose: helping a reader decide whether she would be interested in reading a particular story. They contain just enough information to enable a reader to form adequate expectations about the story, but they do not reveal the plot. According to these criteria, a target summary provides a reader with an idea of whom the story is about, where and when it happens (in a way that goes beyond simply listing names and places) but does not re-tell the events of the story.

In order to build such summaries, the system attempts to identify sentences that meet two criteria: they focus on main entities in the story and they relate the background of the story rather than events. Discussing the criteria for the sentence selection process comprises a large part of this dissertation. These criteria can be roughly divided into two categories: 1) information about main entities (e.g., main characters and locations) and 2) information related to the grammatical aspect of clauses. By relying on this information the system selects sentences that contain important information pertinent to the setting of the story.

Six human judges evaluated the produced summaries in two different ways. Initially, the machine-made summaries were compared against man-made ones. On this account, the summaries rated better than those produced using two naïve lead-based baselines. Subsequently, the judges answered a number of questions using the summaries as the only source of information. These answers were compared with the answers made using the complete stories. The summaries appeared to be useful for helping the judges decide whether they would like to read the stories. The judges could also answer simple questions about the setting of the story using the summaries only. The results suggest that aspectual information and information about important entities can be effectively used to build summaries of literary short fiction, even though this information alone is not sufficient for producing high-quality indicative summaries.

Acknowledgements

This work would not be possible without support and help from a number of people.

Nothing can express my gratitude to my husband, Alex, who patiently listened to my worries and discussed my ideas about language and cognition in general and summarization of fiction in particular. He stoically listened to my presentations and commented on my drafts. Without his support I would not have had the strength to complete this work.

I thank my parents and sister for believing in me even when I did not. I am especially grateful to my father for multiple discussions on the subjects of fiction and language during the preliminary stages of this work. These conversations had an enormous influence on its final shape.

I would also like to thank my supervisor, Dr. Stan Szpakowicz, for setting high standards of quality for scientific research and teaching me how to meet them.

The credit for making my life as a graduate student more cheerful and full of unexpected turns goes to my friends and colleagues, William Elazmeh and Ramanjot Singh Bhatia. Not only did they help by (extensively) commenting on my work at its various stages, but their support and our discussions made these three years interesting and intellectually challenging in ways that go beyond academic research.

I would like to thank Connexor Oy and especially Atro Voutilainen for their kind permission to use the Connexor Machinese Syntax parser free of charge for research purposes. Without this permission, a large portion of this work would not be possible.

Special thanks to Alex Kunadze, Rimma Kazantseva, Ramanjot Singh Bhatia, Scott Briggs, Isha Tan and Catherine Champagne for contributing a lot of time and making evaluation of this work possible.

In addition, I would like to express my gratitude to people who have worked in Natural Language Processing at the School of Information Technology and Engineering. They have created a vibrant intellectual environment that encourages research. I want particularly to thank Oana Frunza, Dr. Diana Inkpen, Dr. Stan Matwin and Dr. Vivi Nastase.

Table of Contents

ABSTRACT I

ACKNOWLEDGEMENTS II

LIST OF FIGURES VII

CHAPTER 1. INTRODUCTION 1

1.1. Text summarization as a research area 1

1.2. Delimiting the scope of the problem 3

1.3. Motivation 4

1.4. Structure of the dissertation 4

CHAPTER 2. TEXT SUMMARIZATION: BACKGROUND AND RELATED WORK 6

2.1. Chapter overview 6

2.2. How humans summarize 6

2.3. Automatic text summarization today 7

2.4. Shallow text summarization 8
2.4.1. Using location in text summarization 9
2.4.2. Using keywords in text summarization 11
2.4.3. Leveraging title words in automatic text summarization 18
2.4.4. Using cue words in text summarization 18

2.5. Leveraging discourse-level structure for text summarization. 19
2.5.1. Using cohesion in text summarization 19
2.5.2. Using coherence in automatic text summarization 22

2.6. Summarization of fiction 27
2.6.1. Overview 27
2.6.2. Understanding fiction 27
2.6.3. Conclusion 30

2.7. Evaluation of text summarization 30
2.7.1. Overview 30
2.7.2. Intrinsic evaluation 31
2.7.3. Extrinsic Evaluation 35

2.7.4. Evaluating the semantic content of a summary 36

2.8. Conclusion 37

CHAPTER 3. OVERVIEW OF THE PROPOSED METHOD 39

3.1. Chapter overview 39

3.2. On the nature of short stories 39

3.3. Corpus description 40

3.4. Defining the objective 41

3.5. Overview of the Approach 42

CHAPTER 4. IDENTIFICATION OF IMPORTANT ENTITIES IN SHORT STORIES. 45

4.1. Chapter overview 45

4.2. Identifying salient entities 45

4.3. The anaphora resolution module 46
4.3.1. The concept of anaphora 46
4.3.2. Tools used in the anaphora resolution module 48
4.3.3. Resolution of pronominal anaphora 50
4.3.4. Resolution of noun phrase anaphora 56
4.3.5. Classifying definite noun phrases 57
4.3.6. Evaluation of the anaphora resolution module 60

4.4. Identification of important characters 61

4.5. Conclusion 62

4.6. Anaphora resolution: related work 62

CHAPTER 5. USING ASPECT TO IDENTIFY DESCRIPTIVE SENTENCES 66

5.1. Chapter overview 66

5.2. Aspect: the concept and fundamentals 66

5.3. Properties of a clause that signal its aspectual type 69
5.3.1. Overview 69
5.3.2. Lexical aspect vs. aspect of a clause 69
5.3.3. Tense and grammatical aspect 71
5.3.4. Temporal expressions and aspect 73
5.3.5. Other indicators of aspect 75

5.4. Establishing aspect automatically 76

5.5. Conclusion **78**

CHAPTER 6. THE DESCRIPTION OF THE SENTENCE SELECTION MODULE 79

6.1. Chapter overview **79**

6.2. Overall system design **79**

6.3. Feature selection: description and motivation **81**

6.4. Handling clauses with the verb *have* **85**

6.5. Experiments **86**
6.5.1. Experimental setting 86
6.5.2. Experiments with manually designed rules 87
6.5.3. Experiments with machine learning 88

6.6. Conclusion **89**

CHAPTER 7. EVALUATION OF THE AUTOMATICALLY PRODUCED SUMMARIES 91

7.1. Chapter overview **91**

7.2. Overview of the evaluation procedure **91**

7.3. Creating gold-standard summaries: Task 1 ..... **93**

7.4. Human judgment of computer-made summaries: Task 2 ..... **100**

7.5. Conclusions **102**

CHAPTER 8. CONCLUSIONS AND FUTURE WORK 103

8.1. Contributions **103**

8.2. Shortcomings **104**

8.3. Future work **105**

REFERENCES 107

APPENDIX A. STORIES IN THE CORPUS. 114

APPENDIX B. LISTS OF NOUNS ADDED TO GATE GAZETTEER 116

APPENDIX C. FEATURES USED IN THE COARSE- AND THE FINE-GRAINED CLAUSE REPRESENTATIONS 118

APPENDIX D. TEMPLATES FOR CAPTURING TEMPORAL EXPRESSIONS.....122

APPENDIX E. PSEUDO-CODE FOR MANUALLY COMPOSED RULES.....128

List of Figures

Figure 2.1. A sample sentence and a corresponding event graph (Lee et al. 2006)……..14

Figure 2.2. An example of a cohesion graph (Mani and Bloedorn1999)…….…….21

Figure 2.3. Example of an RST tree from (Mann and Thompson 1987, p. 51)……23

Figure 2.4. An example of an RST-tree for an article (Marcu 1998)……25

Figure 2.5. Examples of primitive plot units (Lehnert 1982, p.380)…….…...…...….28

Figure 2.6. A complete plot-unit representation of a story (Lehnert 1982, p. 389)……...29

Figure 2.7. Example factoids (van Halteren and Teufel 2003)……..36

Figure 3.1. An example of a manually created summary……..43

Figure 4.1. An example of the Connexor parser output……48

Figure 4.2. An example of a JAPE rule……49

Figure 4.3. An example of an anaphora-resolved document viewed in GATE…….50

Figure 4.4.a. An implementation of the pronoun resolution algorithm of Lappin and Leass (1994) (continued in Figure 4.4.b)……..….51

Figure 4.4.b. An implementation of the pronoun resolution algorithm of Lappin and Leass (1994) (continued from Figure 4.4.a)……..52

Figure 4.5. Noun-phrase anaphora resolution……58

Figure 5.1. Aspectual classification hierarchy (Huddleston and Pullum 2002, p. 118.)...67

Figure 5.2. Types of multiple situations (Huddleston and Pullum 2002, p. 123.)…….68

Figure 6.1. System architecture…….……..……...80

Figure 6.2. Pseudo-code for determining the type of have-clauses based on the WordNet category of direct object (Siegel 1998b.)……..……...86

Figure 6.3. Examples of manually composed rules……. ……....88

Figure 7.1. Example of a summary produced by the system……....92

Figure 7.2. Fragments of summaries produced by 3 annotators for The Cost of Kindness by Jerome K. Jerome……...….……95

List of Tables

Table 2.1. Structure of a typical judgment (Farzindar and Lapalme 2004)……..11

Table 2.2. Modified greedy algorithm (Filatova and Hatzivassiloglou 2004)……..13

Table 2.3. An example of a centroid for a cluster on the topic "Algerian terrorists threaten Belgium" (Radev et al. 2004, p.925)……..……..16

Table 2.4. Features representing a topical segment (Zhou and Hovy 2005)…….17

Table 4.1. Salience weighting factors used for ranking candidate antecedents…….53

Table 4.2. Heuristics for identifying new discourse entities……..60

Table 4.3. Results of anaphora resolution……..61

Table 5.1. Privative featural identification of aspectual classes (Dorr and Olsen 1997)..71

Table 5.2. Types of temporal expressions (Harkness 1987)……..73

Table 6.1. Description of the features in both datasets……..82

Table 6.2. Decrease in F-score caused by removing one feature at a time……90

Table 7.1. Inter-judge agreement……93

Table 7.2. Sentence overlap between computer- and human-made summaries. Majority gold-standard……..……..…..96

Table 7.3. Sentence overlap between computer- and human-made summaries. Union gold-standard…….……..96

Table 7.4. Sentence overlap between computer- and human-made summaries. Intersection gold-standard……..97

Table 7.5. Answers to factual questions……..……..100

Table 7.6. Answers to subjective questions……...……101

Chapter 1. Introduction

1.1. Text summarization as a research area

Despite apparent clarity, the term *text summarization* can be interpreted in many different ways. In the community of researchers studying Natural Language Processing (NLP) it means the task of reducing a coherent and meaningful piece of text to a shorter version in a way that preserves the important information and discards information of secondary importance. This task is accomplished in an automatic manner, using a computer program.

The notion of importance, or *salience*, is crucial when talking about text summarization. *Salience* is "the weight attached to the information in a document, reflecting both the document content as well as the relevance of the document information to the application" (Mani 2001, p. 11). What is important in a document? The answer is that it depends. Among the common important factors are the nature of the audience and the task for which the summary is intended. The audience may have a need for general information about the source documents (*a generic summary*). They may also want to know some specific information (how does this paper advance over the-state-of-the-art technology) and, therefore, require a *query-specific summary*. The audience may consist of specialists in a particular area or of laymen. One may also distinguish between indicative and informative summaries. *An indicative summary* provides a reader with an idea as to what the original is about without giving any details. This type of a summary is useful for initial selection of documents for further inspection. *An informative summary* contains more in-depth information about the original and provides a reader with specific salient points found in the original. All these and many more factors influence what is salient in a particular document within a particular context.

The state-of-the-art of the NLP technology is such that there exists a fair number of syntactic, morphological and shallow semantic tools[1] that are both reliable and reasonably available. On the other hand, deep wide-coverage semantic analysis remains an unaffordable luxury for the vast majority of researchers and industrial enterprises. Because of this, the summarization com-

[1] By shallow semantic tools I mean such tools as gazetteers, named-entity recognizers, *etc*.

munity faces a dilemma: how can one get at the meaning of a document armed with an arsenal of tools capable of analyzing texts only in a shallow manner? The answer provided so far by the scientific community is to approximate meaning using surface information about texts.

The genre that the summarization community explores most frequently is daily news and news wire. Then there are scientific papers, medical documents and legal documents[2]. Creating high-quality summaries for any of these genres is a formidable challenge in itself and continues to be a vibrant research area. Yet, these genres share certain common characteristics. Namely, each of them is associated with a rather rigid, well-established structure (introduction, review of related work, description of work, conclusion, *etc*.). The authors of such documents strive to achieve clarity and coherence and attempt to avoid ambiguity. These traits have been much exploited in automatic text summarization.

On the margins are less orthodox genres: internet chats, dialogues and fiction. These data do not possess characteristics that made summarizing structured documents feasible. Since this dissertation concentrates on summarizing fiction, I will use it as an example. In fiction, writing in accordance with a template (which is synonymous with having an established structure) is a sure way to write poorly. As if the absence of structure was not a challenge enough, other phenomena come into play: the use of metaphor and dialogue, imitation of ungrammatical speech, leaving things unsaid and many other devices that, in fact, make good literature good. Creating summaries of such texts automatically is considerably more difficult. In fact, it has not really been tackled since a few earlier semi-automatic approaches (Charniak 1972; Lehnert 1982).

This dissertation describes an effort to fill this gap. While working on summarizing short stories, I tried to answer several questions:

What constitutes a good summary of a literary work?

What can be done using tools and technologies available today? Is it feasible?

How can one build a summary of a short story?

I hope that this dissertation sheds some light on these questions, if only partially.

[2] For a thorough review of the most important trends in text summarization the reader is referred to **Chapter 2 (Text Summarization: Background and Related Work)**.

1.2. Delimiting the scope of the problem

As the reader might have gathered from **Section 1.1**, the precise meaning of the term *summarization* depends on the context. In particular, it depends on the intended use of the summary. It is, therefore, important to define what the term means in the context of this work and to delimit the scope of the problem.

The objective of this project is defined as follows: given a short story, produce a summary of it such that it contain enough information to enable a reader to decide whether she would like to read the original. Summarizing what exactly happens in the story (*e.g.*, summarizing the plot) is outside the scope of this project. In fact, summarizing the plot is undesirable for the purposes of this project. The summary consists of sentences extracted from the original according to certain criteria.

There are two reasons for excluding the plot from the summary. First, given the objective of helping a reader make informed decisions with respect to the story, revealing the plot in undesirable because it would spoil the discovery. Second, since the problem of summarizing fiction is largely unexplored, summarizing the setting of a story and leaving the plot alone for the moment seems to be a good starting point. Including the plot into the summary would make this already challenging problem even more challenging.

It is also necessary to explain what I mean by a *short story*. The precise nature of the data is described in **Section 3.2** and a complete list of the stories in the corpus is provided in **Appendix 1**. The corpus consists of short stories 2 to 10 pages long, written by mainstream authors mostly in the XIX and early XX century. I did not include experimental fiction (for instance, stories written in the stream-of-consciousness style), and tried not to include stories with excessive amount of dialogue. The stories can be described as social fiction, with the exception of a few fairy-tales.

In the end, starting this project, I had hoped to produce generic indicative summaries of short stories with a specific objective of providing a reader with adequate expectations about the original.

1.3. Motivation

Why would one put any significant effort into summarizing fiction when commercial advantages are more distant than, for instance, those of summarizing legal documents? There are several reasons. The goal of text summarization in general is to help a human reader navigate the vast amounts of data available electronically. Similarly, summarization of fiction aims to make large amounts of literature available online more accessible to readers. Several large online (literary) libraries already exist (e.g., Project Gutenberg http://www.gutenberg.org) and even larger ones are being constructed (e.g., Google Library (Graham 2004)). It seems to me that such a facility would be useful in this context.

In addition, one needs to look at the issue in perspective: textual data available electronically do not consist solely of news articles and hard scientific facts. Continually restricting advances in text summarization to these highly structured genres simply does not give an adequate view of what language is. From this point of view, summarizing any other realistic data type (speech, dialogue, e-mail, personal web pages) is a plus, as it contributes to the understanding of the phenomenon in general.

The data used in the experiments are short stories written during the XIX and the early XX century that might be called "conventional". Why would one summarize short stories when they are, by definition, already short? First, the genre is a more uniform genre of literature than, for instance, novels. Second, given the exploratory nature of this project and utter unavailability of data for this kind of research, I had to do a lot of things from scratch. The small size of short stories made compiling the corpus, annotating it, performing the computations and evaluation feasible. This might not have been the case had longer works been involved. And yet, I hope that literary short stories are as valid a representative of fiction as a genre as are any other and that exploring summarization of short stories will shed some light on other types of literature.

1.4. Structure of the dissertation

This dissertation is organized according to the description below.

Chapter 2 is a brief overview of recent achievements by other researchers in the area of text summarization. It describes important trends and directions and, also, established practices of evaluating summaries. **Chapter 3** introduces the methodology proposed in this dissertation. **Chapter 4** describes the pre-processing stage of the project and **Chapter 5** provides the linguistic motivation for the criteria, which serve as a basis for sentence selection. Since these two chapters (**Chapter 4** and **5**) touch upon two distinct research areas of computational linguistics (namely, anaphora resolution and automatic detection of the grammatical aspect) an overview of related work for each of these areas appears at the end of the respective chapter. **Chapter 6** describes the process of sentence selection and summary construction. **Chapter 7** describes how the summaries are evaluated and how well they rate using chosen criteria. **Chapter 8** concludes by stating the contributions of this work and outlining possible directions for future work.

Chapter 2. Text Summarization: Background and Related Work

2.1. Chapter overview

This chapter provides a brief overview of the body of research in the area of automatic text summarization. I concentrate on single-document summarization and only touch upon multi-document summarization techniques in as much as they are relevant to single-document summarization. The reason for this decision is that the area of multi-document text summarization poses its unique challenges and opportunities, which are not found in single-document summarization.

I begin by listing psychological experiments that attempted to establish how humans produce summaries (**Section 2.2**). An overview of existing approaches to summarizing text follows (**Sections 2.3-2.5**). **Section 2.6** reviews a few distinct approaches to summarization of fiction. **Section 2.7** provides an overview of accepted evaluation procedures for automatically produced summaries.

2.2. How humans summarize

As in many other sub-areas of Natural Language Processing, it is worth studying how humans perform a task that a researcher is trying to accomplish automatically. A number of researchers in different disciplines attempted to study how humans create summaries of various textual genres (in most cases, texts were articles). Teun van Dijk (1979) reports experiments in which students were asked to summarize a literary work presented as a set of propositions. Liddy (1991) establishes what components are essential in an abstract written by a professional summarizer (the abstracts were those of scientific papers). But the most detailed account of how humans summarize is rather recent – it is presented by Brigitte Endres-Niggemeyer (1998).

Endres-Niggemeyer (1998) offers a detailed account of how six professional summarizers produce summaries of articles. She observes three stages that are common to all six subjects:

1. Exploring a document and classifying it using previous knowledge about document types and their structure. During this stage a summarizer establishes familiarity with the

document and evokes a *scheme* – a mental representation of his expectations about the document type and the structure. A different scheme is evoked for different document types.

2. Finding relevant information. Endres-Niggemeyer (1998, p.150) calls this stage "solving aboutness problem". During this stage a summarizer skims a document looking for cues (or hints) that signal important information. A summarizer may read the opening and/or closing paragraphs of the document or its sections, look for words occurring in the title or for other emphases. Endres-Niggemeyer (1998, p.152) lists several means to achieve emphasis: *a privileged position* in the title or the opening of a text component, *special layout features* (font, colour), *cue phrases* (*e.g.*, *first and foremost*), *repetition* and *rephrasing*. She argues that a summarizer constructs a mental representation of what a document is about called a *theme*.

3. Composing a summary. A summarizer attempts to follow the author as closely as possible and produces an abstract by cutting and pasting the text from the original and then smoothing it. This is in line with the results reported by Jing and McKeown (1999) who closely analyzed 15 manually written summaries and found that 78% of their sentences were a result of cutting and pasting from the original documents.

It is interesting to note that professional abstractors do not attempt to read the source document from the beginning to the end. Instead, they skim the document for hints as to what it is about and then construct a summary by cutting and pasting. Such a strategy appears hopeful for automatic text summarization since it suggest that it is not necessary to understand a document completely in order to summarize it. As we will see in **Sections 2.4 – 2.5**, most automatic systems try to leverage at least one of the cues suggested by Endres-Niggemeyer (1998).

2.3. Automatic text summarization today

The majority of automatic summarizers implemented until now are extraction-based. This means that a summarizer selects textual units (usually sentences, but paragraphs and topical segments are also possible) from the original document based on a scoring function and then arranges them into a summary as they are or with relatively little editing. There are exceptions to this statement (Barzilay and McKeown 2005; McDonald 2006) but I do not discuss them in this dissertation. Abstracting (as opposed to extracting) consists of two stages: identifying salient

portions of the original document and paraphrasing the source text into the ultimate summary. The first stage is the similar for both abstractive and extractive summarizers. The second stage is unique to abstracting, but it is closer to Natural Language Generation than to text summarization. This is why I omit abstracting in this review.

The summarization systems can be roughly divided into two categories: systems that rely on shallow processing and those that use the informational structure of documents. The systems falling into the first category identify salient passages in the original documents by relying on the presence or the distribution of various surface markers of salience[3]. The second group of systems models the informational (or discourse) structure of a document and then leverages to find salient information. **Section 2.4** contains an overview of shallow approaches and reviews a set of techniques that are commonly practiced. **Section 2.5** is a review of the systems that use document structure.

2.4. Shallow text summarization

Most text summarization systems proposed until now rely on identifying surface markers of salience in documents and use their distribution to find important information. Although there are many systems, the fundamental tools that they use can be divided into a small number of categories. In fact, one can identify four underlying types of surface markers of salience on which such systems rely (Mani 2001): keywords, cue words, title words and position in the document.

Keywords are a set of words that conveys the central theme of a document. Keywords can be identified in several ways: these may be the most frequent words in a document (Edmundson 1969), they may be the words with the highest *tf.idf* scores[4] (Radev et al. 2004) or they may be selected using a keyword extracting software (Copeck et al. 2002). A few notable keyword-based approaches are reviewed in **Section 2.4.2**.

[3] In this dissertation I refer to this group of approaches as *shallow* approaches. I use this term for brevity. It does not reflect in any way the thoroughness of these approaches or the depth of processing that the source documents undergo.

[4] *tf.idf* score is explained in **Section 2.4.2**.

Cue words are words and expressions that signal that a sentence is relatively salient, or, on the contrary, that it is unimportant. Examples of such expressions include *most importantly, to conclude, finally, first and foremost, etc.* Most approaches that rely on cue words are discourse-based approaches that are reviewed in **Section 2.5**. The reader will also find a few examples of using cue words in combination with other salience markers in **Sections 2.4.1 - 2.4.2**.

Location of a sentence is generally considered a good indicator of its salience. As a rule, sentences occurring in the opening paragraphs of factual documents or of their sections tend to be more salient than sentences found in other locations (Lin and Hovy 1997). In fact, in genres such as newswire, the first paragraph of a document is its excellent summary. In addition, sentences opening and closing separate paragraphs also tend to be of special importance. See **Section 2.4.1** for a review of a few exemplary approaches.

Nor surprisingly, document and section titles usually give a good indication of what the document is about. This is why sentences that contain many words found in the titles tend to be informative. Using title words for text summarization is described in **Section 2.4.3**.

It is interesting to note that the surface markers of salience leveraged by the summarization systems overlap significantly with the markers of emphasis described by Endres-Niggemeyer (1998). In the remainder of **Section 2.4** I will review a number of systems that rely on these markers of salience to identify salient passages in documents. I attempt to group the systems so as to exemplify each marker type, but very often a system combines a number of distinct approaches. In such cases, I concentrate on a specific type of the salience marker and list others just enough to allow general understanding of the system.

2.4.1. Using location in text summarization

A work that examines in detail the relation between the position of a sentence in the document and how much information it bears is Lin and Hovy (1997). Lin and Hovy (1997) examine 13,000 newspaper texts about computers, each of which is accompanied by a list of relevant keywords and an abstract. They approximate the informativeness of individual sentences and paragraphs by the number of keywords they contain and by how much they resemble the accompanying abstract. The results suggest that the first 1-3 paragraphs of an article are most informa-

tive and that within each paragraph early mentions of sentences are more informative than subsequent ones. It must be borne in mind, however, that these findings would probably differ for other text genres.

The position of a sentence in a document may be revealing in a way that goes beyond simply selecting the opening sentences and paragraphs. Certain types of documents exhibit a very rigid structure, which predefines the location of important information. PERSIVAL, a customizable summarizer of medical documents (Elhadad et al. 2005) leverages this property of medical articles to summarize them. PERSIVAL creates summaries of medical journal articles suitable for the needs of either patients or doctors (depending on who is using the system)[5]. Given a query (*e.g.*, *What are possible treatments of pneumonia?*) and a patient's record to serve as the context, PERSIVAL returns a summary suitable for a given user. During the first step, information retrieval techniques are used to retrieve a set of articles that are likely to be relevant. (Elhadad et al. 2005) report that clinicians can quickly determine relevance of an article by skimming through its **Results** section. Using this knowledge, (Elhadad et al. 2005) leverage the rigid structure of medical articles and only work with **Results** section of any candidate paper. They employ text categorization techniques and pattern matching to select sentences that contain results relevant to caring for patients.

Another example of a genre characterized by rigid structure is legal documents. Systems that summarize such documents tend to make heavy use of their structure. One example of such a system is LetSum (Farzindar and Lapalme 2004)[6]. LetSum is a system for summarizing the judgments of the Federal Court of Canada. Farzindar and Lapalme (2004) have observed that a typical judgment in their corpus has a certain organizational structure. Namely, each judgment has elements shown in **Table 2.1**. Before producing a summary LetSum identifies portions of a document corresponding to each element. It does so by using a heuristic that relies on identifying informative section titles and the position of textual units within a document (*e.g, Conclusion* is likely to appear at the end of the document, while *Introduction* - at the beginning). The heuristic also makes use of the amount of direct speech found in a passage and the presence of certain lin-

[5] The system also creates summaries of video and audio material, but these capabilities are not reviewed in this dissertation.
[6] Another similar system is (Teufel and Moens 2002) described in **Section 2.5.2**.

Table 2.1. Structure of a typical judgment (Farzindar and Lapalme 2004).

Thematic structures	Content	Judg-ment	Sum-mary
Decision data	Name of the jurisdiction, place of the hearing, identity of the author, names of the parties, title of proceeding and authority and doctrine		
Introduction	Who did what to whom	5%	12%
Context	Facts in chronological order or by description	24%	20%
Juridical analysis	Comments by judge, finding of facts and application of the law	67%	60%

guistic markers (in the context of this work the linguistic markers are words such as *summarize, conclude, etc.*)

Once the segmentation is complete, LetSum selects the most informative sentences from each segment. In order to identify such sentences, the system favors those that occur in the opening paragraphs of a document, of their parent segment and those that occur early in paragraphs. Other measures of informativeness include high *tf.idf* score (see **Section 2.4.2** for description of *tf.idf)*. LetSum combines these measures into a heuristic function and arranges the most informative sentences into a table-like summary.

2.4.2. Using keywords in text summarization

Key words and key phrases have been perhaps the single most exploited feature for extractive summarization. Intuitively, textual units that contain many important words/phrases are likely to be salient. Exactly which words are important varies depending on the context and the system. In a very classical work on text summarization Edmundson (1969) defined keywords as the most frequent words in a document excluding a number of manually selected stop-words. Copeck et al. (2002) use three different automatic key phrase extractors to identify terms that are good representatives of a text at hand (these extractors are KEA (Witten et. al. 2002), an extractor described in (Turney 2002) and NPSeeker, a University of Ottawa's program).

(Zechner 2002) describes a system for summarizing transcripts of domain-independent dialogues. (Zechner 2002) discusses many challenges pertinent to summarization of transcripts in general (such as transcription errors) and to summarization of dialogues in particular (interdependence of questions and answers). After several pre-processing steps (sentence-boundary detection, POS tagging, *etc*.), each dialogue is split into topical segments. The system then computes a lexical similarity measure between a vector of words representing the whole segment and each sentence. Sentences with high similarity with the segment are rewarded. Although there is no explicit use of key phrases within this system, it implicitly favors sentences that contain more words representative of a segment.

Among the systems that heavily leverage frequent words and phrases found in a document are so-called *event-based summarizers* (Filatova and Hatzivassiloglou 2004; Li et al. 2006). Instead of simply considering the most frequent words as most important ones, these systems concentrate on identifying atomic events that are central to a document (both systems reviewed in this section are tailored for summarizing news articles). An atomic event consists of an action and its main constituents or actors.

Filatova and Hatzivassiloglou (2004) identify atomic events by finding all pairs of named entities or top ten most frequent nouns that co-occur within a single sentence. The authors call such pairs *relations* and all words occurring between them - *connectors.* The system filters out the unimportant connectors, only preserving those that are verbs and action nouns according to WordNet ontology (Fellbaum 1998). Eventually, the list of events contains only triplets consisting of a relation and a connector thus defined. In this representation of atomic events, the connector is the action and the relation is the actors or the constituents. The authors rate salience of events according to the following formula:

$$S_{event} = S_{relation} * S_{connector}$$

where

$$S_{connector} = \frac{occurences\_of\_the\_current\_connector}{total\_number\_of\_connectors}$$

$$S_{relation} = \frac{occurences\_of\_the\_current\_relation}{total\_number\_of\_relations}$$

The system then selects the most informative sentences and composes a summary out of them: the score of a sentence is the sum of scores of the events that it contains. The optimal algorithm for sentence selection proposed by the authors is shown in **Table 2.2**.

A development along the same line of research is presented by Li et al. (2006). Unlike the system of Filatova and Hatzivassiloglou (2004), Li et al. (2006) rely on event-based summarization for producing summaries for a collection of documents that focus on the same topic. They use the corpus that was released for Document Understanding Conference in 2001. This system builds a graph of atomic events for a document and then identifies the most salient nodes in the graph. Similarly to Filatova and Hatzivassiloglou (2004), the authors view an event as an action accompanied by its constituents. However, in their view, time and place of an event are also its integral constituents, whenever such information is available. In order to build an event graph for a document, the system identifies four types of named-entities: organization, person, location and date (this is achieved using GATE software (Cunningham et al. 2002)). It also identifies action verbs and nouns that connect such entities; the authors refer to these connectors as event terms (ET) (see **Figure 2.1** for an example of this representation).

The next stage involves identifying the most salient nodes in the graph. In order to do that, the system computes the strength of a connection between any two connected nodes in the graph in terms "of relevance defined from different perspectives" (p.371). Let $w(\ node_i\ )$ be the salience of a $node_i$ and $r(\ node_i,\ node_j\ )$ be the strength of connection between $node_i$ and $node_j$. Then $w(\ node_i\ )$ is calculated using the formula below and PageRank algorithm (Page et al. 1998):

Table 2.2. Modified greedy algorithm (Filatova and Hatzivassiloglou 2004).

1. Calculate the score of every sentence as the sum of the scores of all events it covers.
2. Only consider sentences that contain events with the highest weights that have not yet been covered by the summary. Out of these, choose the highest-ranking sentences and add them to the summary. Add newly covered events to the list of covered events.
3. Reweigh the sentences: subtract from each sentence's weight the weight of the events that have already been covered in the summary.
4. Go back to step 2 if the summary is shorter than the desired length.

$$w(node_i) = (1-d) + d\left(\frac{w(node_1)}{r(node_i, node_1)} + \ldots + \frac{w(node_j)}{r(node_i, node_j)} + \ldots + \frac{w(node_t)}{r(node_i, node_t)}\right)$$

where $node_j$ ($j = 1, 2, \ldots, t;\ j \neq i$) are the nodes connected to $node_i$ and d is a constant factor set to 0.85.

The authors propose several ways to compute the strength of a connection between any two nodes (they refer to it as *relevance*). They distinguish between intra-event relevance and inter-event relevance. The intra-event relevance measures how strongly an action of an event is associated with its constituents (*i.e.*, the strength of a connection between a event term (ET) and named-entity nodes (NE) in the same event) and it is denoted *R(ET, NE)*. Inter-event relevance measures the strength of association between any two event terms (ET) or between any two named-entity nodes (NE) that are not part of the same event. This type of relevance is denoted *R(ET, ET)* or *R(NE, NE)*.

Figure 2.1. A sample sentence and a corresponding event graph (Lee et al. 2006).

<Organization> America Online </Organization> was to buy <Organization> Netscape</Organization> and forge a partnership with <Organization> Sun </Organization>, benefiting all three and giving technological independence from <Organization> Microsoft</Organization>.

The inter-event relevance reflects how many times an event term et_i co-occurs with a named-entity ne_j:

$$r(et_i, ne_j) = freq(et_i, ne_j)$$

The authors propose a number of ways to measure inter-event relevance. They measure R(ET, ET) in two ways: by using the similarity package WordNet::Similarity (Pedersen et al. 2004) or by measuring how many times any two events terms share the same constituents. The motivation behind the second method is that events with the same constituents are likely to be related:

$$r(et_i, et_j) = \left| NE(et_i) \cap NE(et_j) \right|$$

Inversely, the relevance of two named-entities can be measured in the same manner:

$$r(ne_i, ne_j) = \left| ET(ne_i) \cap ET(ne_j) \right|$$

Li et al. (2006) measure inter-event relevance in two additional ways: by rewarding co-referring named entities and by rewarding named entities that co-occur within a window of pre-specified size.

Once the strength of all connections is known, PageRank re-weighs the graph and makes it possible to identify the highest-ranking nodes. The significance of sentences is then obtained from the significance of the nodes that it contains.

A different approach that heavily relies on the representative words in a document is the centroid approach (Radev et. al. 2004). It is different from the systems reviewed so far in that it can be used exclusively for summarizing collections of related documents (*i.e.* it is not suitable for single-document summarization). It was designed and tested using collections of newswire articles. The approach combines two distinct systems: CIDR (Radev et. al. 1999) and MEAD (Radev et. al. 2004). CIDR is responsible for creating clusters of documents that focus on the same topic while MEAD selects summary-worthy sentences and combines them into a summary.

In order to create clusters of related documents CIDR uses modified *tf.idf* score to cluster articles according to their central topic. Originally, *tf.idf* (Salton and Buckley 1988) is a measure that ranks sentences according to their importance in a way that rewards sentences that contain words that are frequent in a document but are infrequent in the corpus. *tf.idf* score can be computed using the following formula:

$$ds_{ij} = \sum_{k=1}^{n_{si}} tf_{jk} \cdot \log(\frac{n_d}{df_k})$$

In this formula, ds_{ij} the *tf.idf* score of sentence i in a document j, n_{si} is the number of words in sentence i, k is the *kth* word in the sentence i, tf_{jk} is the frequency of the word k in the document j, n_d is the number of documents in the corpus and df_k is the number of documents in the corpus that contain the word k.

CIDR uses a modified version of this score to create clusters of documents. A document is considered to be relevant to a cluster if its *tf.idf* score is above a predefined threshold. "A centroid is a group of words that statistically represent a cluster of documents." (Radev et al. 2004, p. 920) At first, CIDR generates a centroid using the first available document (by selecting words with the highest *tf.idf* score). As each subsequent document becomes available, it joins the cluster if its *tf.idf* score is close to that of the centroid (else it is discarded). A sample centroid for a cluster "Algerian terrorists threaten Belgium" is displayed in **Table 2.3**.

Once the cluster is created and its centroid is known, MEAD assigns a salience score to each sentence of every document. The score is computed according to the following formula:

$$Score(sent_i) = w_c C_i + w_p P_i + w_f F_i - w_r R_s$$

C_i is the centroid score, which is the sum of *tf.idf* scores of all centroid words present in a sentence:

$$C_i = \sum_w C_{w,i}$$

P_i is the positional value, which favors sentences that occur early in the documents. F_i is reflects word overlap between sentence *i* and the first sentence in a document. Weights w_c, w_p and w_f are constant and equal. The term $w_r R_S$ penalizes redundancy by reducing the score of sentences that significantly overlap with other sentences with higher *tf.idf* scores.

The sentences with the highest scores are then extracted and combined into a summary.

Table 2.3. An example of a centroid for a cluster on the topic "Algerian terrorists threaten Belgium" (Radev et al. 2004, p.925).

Term	tf	idf	Tf.idf
Belgium	5.5	5.6	30.81
Islamic	3.0	9.80	29.42
GIA	7.0	3.0	21.00
Arabic	1.50	9.11	13.67
jailed	2.00	6.76	13.52
AI	1.50	7.17	10.75
Hardline	1.00	9.81	9.81
Statement	2.50	3.84	9.61
Torture	1.00	8.42	8.42
Threat	1.50	5.44	8.15

Zhou and Hovy (2005) propose a system for summarizing technical Internet Relay Chats (IRCs) on the topic of GNUe development. The authors point out that IRCs differ from more conventional data types in several respects. The chats in their corpus are technical in nature and tend to consist of a problem initiating segment (*i.e.*, someone asking a question) and many response segments. Most messages contain in-depth discussion of the issue and are rather lengthy. In addition, a particular chat may deal with more than one issue (*i.e.*, a person may ask more than one question). The authors refer to this issue as *subtopic structure.*

Table 2.4. Features representing a topical segment (Zhou and Hovy 2005).

-number of overlapping words
-number of overlapping content words
-ratio of overlapping words
-ratio of overlapping content words
-number of overlapping tech words
-the number of messages between the initiating and the responding segment

Due to these particularities, the authors suggest that it is necessary to segment the messages and group them according to sub-topics. The system proceeds by splitting each chat into topical segments using TextTiling (Hearst 1997). Once the segmentation is complete, the segments are clustered according to their subtopic. This is achieved by applying hierarchical agglomerative clustering using *tf.idf* score of each word in a segment. At the end of this process the authors have the chats divided into topical segments where each segment is associated with a particular subtopic cluster. Each cluster has an initiating segment (the segment that occurs early in the chat) and responding segments.

In order to produce summaries, the authors extract the problem initiating segment and then attempt to identify the most informative response segment. They use two methods to do so: Maximum Enthropy (ME) (Berger et al. 1996) and Support Vector Machines (Cristianini and Shawe-Taylor 2000). Regardless of the method, each segment is represented as a vector of structural and lexical features presented in **Table 2.4**.

As the reader might have noticed, the keyword-based methods are an umbrella uniting a variety of distinct and very different views as to what words are representative of a document and how they can be used to produce high-quality summaries. They are similar in that they rely on identifying a set of important words in a document and using their distribution to find salient passages. This group of approaches has witnessed a lot of development in the last five years and it appears to be a dominant group in the summarization community.

2.4.3. Leveraging title words in automatic text summarization

Many summarization systems capitalize on the fact that document and section titles are, in general, informative. This is especially true for factual genres, such as newswire and scientific articles. However, it is uncommon for a title to be informative enough to suggest the gist of an article by itself or, for that matter, to be the only guide of a text summarization system. This is why, although many systems utilize title words for text summarization, I am not aware of any that relies exclusively on this information.

Several systems reviewed in **Section 2.4.2** make use of informative titles. LetSum uses section titles to help identify topical elements of court judgments. PERSIVAL identifies *Results* sections of medical articles in order to restrict summaries to information that is most likely to interest the clinicians. One of the founding works on automatic text summarization (Edmundson 1969) also made heavy use of title words. H.P. Edmundson analyzed 200 scientific papers in chemistry attempting to find out what sentences constituted a representative extract of an article. He represented each sentence as a vector of the following features: cue words, keywords, location of a sentence and title words (*i.e.* words found in the document title and section headings). Using these features, a training and a test set of documents and a linear regression function, Edmundson found that a combination of cue words, title words and location features (with manually assigned weights) yielded the best summaries for his corpus.

The work of Teufel and Moens (2002) reviewed in **Section 2.5.2** also makes use of this feature.

2.4.4. Using cue words in text summarization

Just as it is the case with title words, cue words alone are not informative enough to guide a summarization process. However, the cue words are often used in the approaches that attempt to establish the informational structure of documents. Such approaches are reviewed in **Section 2.5.2**.

2.5. Leveraging discourse-level structure for text summarization.

A document in the common sense of the word is more than a collection of concatenated sentences - it has an overall structure, referred to as discourse structure. Discourse structure is the informational structure of a document. It determines the organization of a document and how the task of conveying a message is distributed across the whole text.

Experiments conducted by Endres-Niggemeyer (1998) suggest that people rely heavily on their knowledge of discourse structure when searching a document for pertinent information. As I have mentioned in **Section 2.2**, according to (Endres-Niggemeyer 1998), humans begin the summarization process by evoking an appropriate *scheme* of a document. A scheme represents their expectations about its type and structure. During the next step, human summarizers quickly look through a document and build a *thematic representation* of it – a picture of what the document is about. During this step summarizers also link the main points to the corresponding passages in the text, thus building an internal discourse representation of a document.

A number of automatic summarizers attempt to construct a representation of the discourse structure of a document and then employ it to find salient information. (Mani 2001) identifies two distinct groups of approaches that employ discourse-level information in text summarization and I follow his classification in this dissertation.

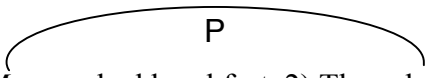
The first group of approaches relies on the phenomenon of *cohesion* (Halliday and Hasan 1996). Cohesion involves relations between words, word senses, or referring expressions, which determine how tightly connected the text is (Mani, 2001, p. 92). A few exemplary approaches in this direction are discussed in **Section 2.5.1**.

Another group of approaches relies on the phenomenon of *coherence*. Coherence deals with inter-relations of textual units at a higher level – relations between sentences and paragraphs and how they form a sensible (*i.e.*, coherent) structure. Such approaches are described in **Section 2.5.2**.

2.5.1. Using cohesion in text summarization

Cohesion describes how many explicit linguistic connections exist between textual units. The connections that determine how cohesive of a text is include repetition of words, use of ana-

phoric expressions, synonymy, meronymy, hyponymy and some others. As a rule, a text that requires a reader to rely on previous sentences more when reading new ones is more cohesive than a text where each sentence talks about unrelated issues.



Example 2.1 1) Mary$_{ent1}$ had breakfast. 2) Then she$_{ent1}$ went to school.

Example 2.2 1) Mary had breakfast. 2) Then Bob went to school.

Examples 2.1 and *2.2* illustrate the concept. *Example 2.1* is more cohesive than *Example 2.2* because the sentences are linked by the pronoun *she* (a link of type prominalization), which refers to *Mary*. *Example 2.2*, on the other hand, is a perfectly valid and sensible piece of text, yet it is less cohesive because sentences *1)* and *2)* are not linked by any cohesion links (repetition, synonymy, prominalization, *etc*.)

Skorohodko (1972) laid the ground for most approaches using cohesion to find salient units in text. Skorohodko chose sentences to be basic units of his analysis. He considered that two sentences had a semantic link between them if a pair of words in these sentences (one from each sentence) was semantically related in one of the following ways: 1) the words were the same (*i.e.*, repetition), 2) the words were synonyms or hypernyms/hyponyms, 3) the words were among the most salient (in his case, frequent) words in the text and both of them are related to a common third word. (Skorohodko 1972) represented a document as a graph where nodes are sentences and undirected edges are semantic relations between them – a topology, that is common in today's cohesion-based approaches too (Mani and Bloedorn 1999; Mihalcea and Tarau 2005).

Skorohodko identified two cohesion-related criteria that help determine how salient a sentence is. He did not name then explicitly, but Mani (2001, p. 95) names them and I use his terminology here:

The connectivity criterion: The salience of a sentence is proportional to the number of sentences that are connected to it.

The indispensability criterion: The salience of a sentence is proportional to the degree of change in the document graph if the sentence is removed.

The basic idea behind these criteria – the idea that the more semantically connected a sentence is, the more salient it is – has been used in automatic text summarization, as I shall explain below.

The work of Mani and Bloedorn (1999) provides an example. The system is tailored to a particular task: given a user query and a pair of articles, it produces a summary that is relevant to the query (*i.e.,* a query-specific summary). Initially, the system constructs a graph representation for each document (an example of this representation for a particular sentence is shown in **Figure 2.2**). In this representation, each word is a node and an edge is one of the following semantic relations between a pair of nodes:

ADJ: a link that connects textually adjacent nodes (*i.e.*, adjacent words)

SAME: a link that connects two instances of the same word (*i.e.*, repetition)

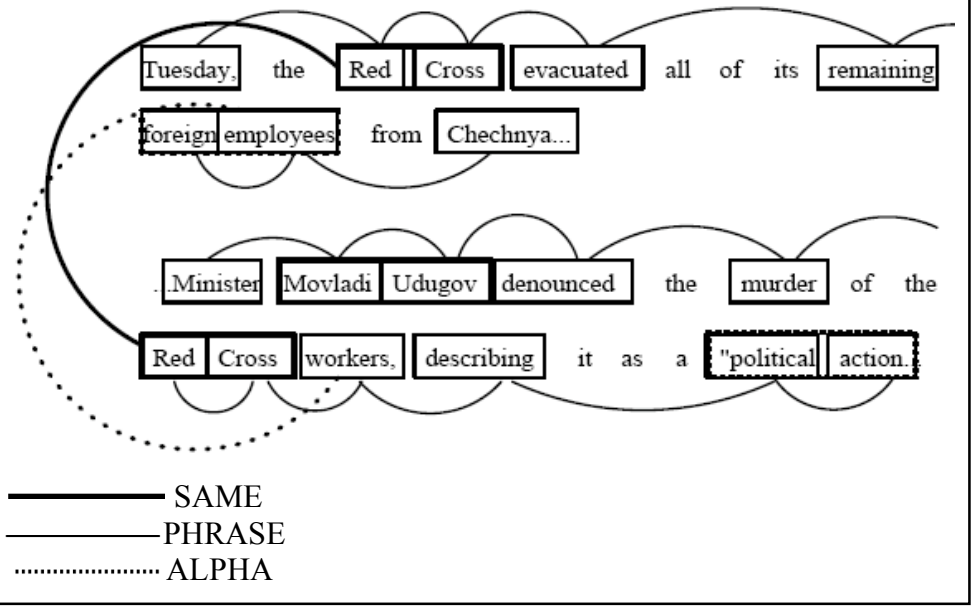
PHRASE: connects adjacent nodes that belong to the same sentence

NAME: connects sub-graphs (strings of words) that are part of the same proper name

COREF: a link that connects sub-graphs (strings of words) that refer to the same named entity.

ALPHA: an edge that exists between two words linked by a synonymy or a hypernymy relation.

Figure 2.2. An example of a cohesion graph (Mani and Bloedorn 1999).



Given a document graph and a user query, the system attempts to find document nodes that match the query. It assigns a weight of 1 to each node selected in this manner and propagates the

weight along the edges that are connected to the node. During the propagation procedure the weight decreases and the rate of decrease is a function of how strong a given edge is: SAME edges are the strongest, followed (in the order of decreasing strength) by NAME, COREF, PHRASE, ALPHA and ADJ edges. After the propagation procedure is complete, a salience contour of a document as it is related to the query emerges. The authors call this process 'finding peaks of salience'.

A more recent work that relies on cohesion to produce summaries is Mihalcea and Tarau (2004). The authors propose a solution to both single- and multi-document summarization, but I only review the single-document summarizer here.

The summarizer proposed by Mihalcea and Tarau (2005) proceeds in the following manner. It begins by building a graph model of a document. The nodes in the graph represent sentences and the edges are inter-sentence connections. The connections are defined as a function of lexical overlap between sentences normalized by sentences' length (therefore, this system only uses repetition relations between sentences). The graph is directed: the incoming edges represent the connections between a sentence and its predecessors (*i.e.*, sentences that occur earlier in the document) and the outgoing edges – between the sentence its successors. During the next step, the system ranks the nodes using two different algorithms: PageRank (Page et al., 1998) or HITS (Hyperlinked Induced Topic Search) (Kleinberg 1999).

Once the ranking has been completed, the system selects sentences with the highest scores for inclusion in the summary.

2.5.2. Using coherence in automatic text summarization

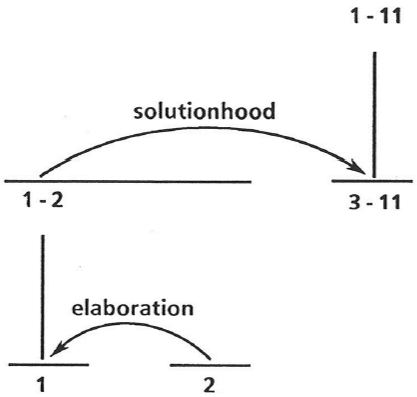
The notion of coherence deals with "the macro-level, deliberative structuring of multi-sentence text in terms of relations between sentences (or clauses)." (Mani 2001, p. 106) Several examples of coherence relations are shown in **Figure 2.3** (the example comes from (Mann and Thompson 1987, p. 51)).

The example in **Figure 2.3** illustrates that a sequence of sentences 1-3 is more than a concatenation of unrelated text: sentence 2 elaborates upon what is expressed in sentence 1, and sentence 3 offers a solution to the problem described in sentences 1 and 2.

A number of theories attempted to explain and describe macro-relations between textual units. (Hobbs 1985) proposes a list of eight coherence relations that can hold between two units of text. The relations are *contrast*, *elaboration*, *evaluation*, *exemplification*, *explanation*, *occasion, parallel* and *violated expectations*. Hobbs attempts to provide a precise semantic definition for each relation, but he proposes no guidelines for identifying them within text. However, the nature of these relations is such that occasionally it is difficult even for a human to distinguish between them, let alone a computer.

Grosz and Sidner (1986) argue that in order to understand the structure of a document one needs to look at its three separate characteristics: *the linguistic structure* of a text, its *intentional structure* and *an attentional state*. *The linguistic structure* refers to the surface structure of a document. It deals with identifying primitive segments of a document and relations between them (the authors do not elaborate upon how such segments are to be identified and upon the types of relations that hold between them). *The intentional structure* has to do with the idea that every (coherent) document has a purpose and so does its every segment. The discourse purpose of a segment is related to how a particular span of text contributes to the overall purpose of the document. As the notion of purpose and intentional structure is related to the author's intentions, there can be any number of such purposes and relations between them, which makes automation of this model very difficult. *The attentional state* is an abstraction of a reader's focus that changes as the discourse unfolds. According to Grosz and Sidner (1986) a grasp of all these components is needed in order to understand the

Figure 2.3. Example of an RST tree from (Mann and Thompson 1987, p. 51).



1. One difficulty … is with sleeping bags in which down and feather fillers are used as insulation.
2. This insulation has a tendency to slip towards the bottom.
3. You can redistribute the filler
4. … 11.

discourse structure of a document.

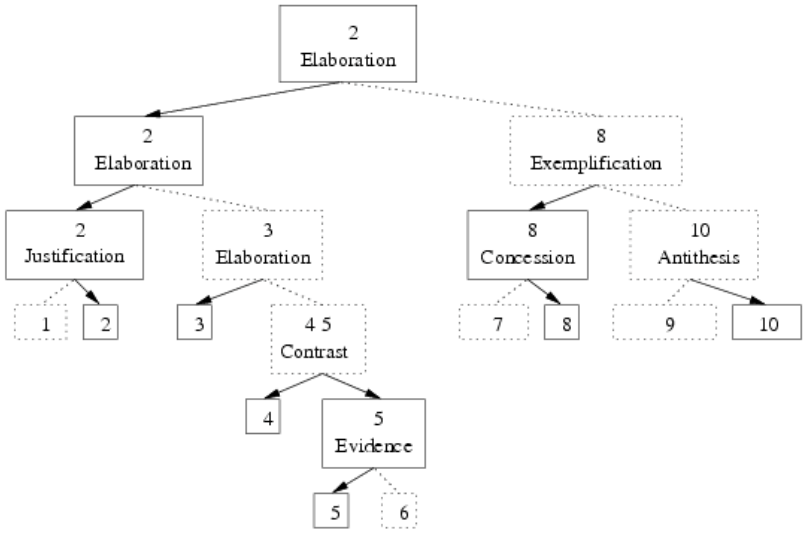
A theory that is perhaps most widely known today and that has been applied automatically is the Rhetorical Structure Theory (RST) proposed by Mann and Thompson (1987). The authors propose a set of 23 possible rhetorical relations that can hold between clauses, sentences, or larger textual units. They highlight that this set is by no means all-inclusive and that it can be further extended. According to (Mann and Thompson 1987), most of the relations are asymmetric, *i.e.,* one unit is more salient than the other. The more salient unit is called a *nucleus* and the less salient one is referred to as a *satellite*. In **Figure 2.3,** the nuclei are nodes towards which the edges (*i.e.,* relations) are directed. The satellites are nodes from which the edges spring. While most of the relations in (Mann and Thompson 1987) are binary, they do not have to be, and ternary and n-ary relations are feasible. The relations also do not have to be asymmetric, although most of them are. Within the Rhetorical Structure Theory, the overall structure of a text is a tree, of a type similar to the one shown in **Figure 2.3**.

A few members of the text summarization community attempted to employ macro-relations within documents in text summarization. The most remarkable of such examples can be found in (Marcu 2000). Marcu extends the RST in several ways: he significantly extends the number of possible rhetorical relations (from 23 to 54) and also proposes and implements a system that automatically constructs RST-trees à la Mann and Thompson. The author manually analyzes 2,100 fragments from the Brown corpus. He selects the fragments so that each fragment contains at least one occurrence of a cue phrase from a list of 450 cue phrases that he composed. From the annotated 2,100 fragments, only 1197 had discourse usage and 1017 were sentential or pragmatic[7]. Marcu annotates each of the 1197 fragments with several pieces of information, such as boundaries of a segment and a cue phrase that introduces it, whether the segment is salient or not (*i.e.,* whether it is a nucleus or a satellite), the type of rhetorical relation it introduces and several others (the complete list can be found in (Marcu 2000, p. 107).) Using this corpus he manually creates a set of rules that guide the automatic construction of RST-trees. He also trains a machine learner to perform the same function.

[7] The numbers do not add up to 2,100 because in several cases cue phrases had multiple roles.

Figure 2.4. An example of an RST-tree for an article (Marcu 1998).

[With its distant orbit --- 50 percent farther from the sun than Earth --- and slim atmospheric blanket, 1] [Mars experiences frigid weather conditions. 2] [Surface temperatures typically average about –60 degrees Celsius (-76 degrees Fahrenheit) at the equator and can dip to -123 degrees C near the poles. 3] [Only the midday sun at tropical latitudes is warm enough to thaw ice on occasion, 4] [but any liquid water formed in this way would evaporate almost instantly 5] [because of the low atmospheric pressure. 6] [Although the atmosphere holds a small amount of water, and water-ice clouds sometimes develop, 7] [most Martian weather involves blowing dust or carbon dioxide. 8] [Each winter, for example, a blizzard of frozen carbon dioxide rages over one pole, and a few meters of this dry-ice snow accumulate as previously frozen carbon dioxide evaporates from the opposite polar cap. 9] [Yet even on the summer pole, where the sun remains in the sky all day long, temperatures never warm enough to melt frozen water. 10]



Summary:

Mars experiences frigid weather conditions. Most Martian weather involves blowing dust or carbon dioxide. Surface temperatures typically average about –60 degrees Celsius (-76 degrees Fahrenheit) at the equator and can dip to -123 degrees C near the poles. Yet even on the summer pole, where the sun remains in the sky all day long, temperatures never warm enough to melt frozen water.

Marcu applies the automatically constructed RST-trees to the task of text summarization. His test-bed consists of five articled from *Scientific American* and 35 articles from the TREC collection (Jing et al., 1998). For each article 13 judges manually build a corresponding RST-tree and manually annotate each primitive RST segment (a leaf of an RST-tree, usually a clause or a sentence) with a salience score.

In order to explain how Marcu uses RST-trees for text summarization, I will use the article shown in **Figure 2.4** as an example. The figure also shows the corresponding tree. Each RST-node is separated using square brackets. All nodes have unique ids (the number before the clos-

ing square bracket). Dotted boxes are satellites and solid-line boxes are nuclei. A dotted-line edge connects a node to its satellite-child, while a solid-line edge connects a node to its nucleus-child. Each parent node has the name of a relation between its children written at the bottom of its box.

We can see from **Figure 2.4** that a nucleus of each relation is a parent for the corresponding sub-tree. For instance, the relation between nodes 5 and 6 is of type *Evidence*, with the node 5 being a nucleus. Consequently, it is promoted to appear as a parent of the sub-tree. Marcu proposes that the salience of a node is related to how high within an RST-tree it is promoted. According to this criterion, node 2 is the most salient in this tree, followed by node 8, then by node 3, etc. The summary obtained by this method is shown at the bottom of **Figure 2.4**.

Thione et al. (2004) describe a similar approach. The system, PALSUMM, depends on a bundle of proprietary software, FX Palo Alto's LInguistic Discourse Analysis System (LIDAS). LIDAS constructs trees that follow the overall structure of texts, but the relations and units of the trees are somewhat different. While RST models the discourse-level relations as edges in the trees, LIDAS uses syntactic relations of types coordination, sub-ordination and n-ary to link basic discourse units, or BDUs. Unlike the RST-trees, the trees constructed by LIDAS have content (*i.e.,* text) in every node, not just the leaf nodes. Once the tree is constructed, the system prunes it at a level that corresponds to the desired compression rate[8].

A slightly different view of the discourse structure and how it can be used in text summarization is proposed by Teufel and Moens (2002). The authors work with a collection of papers on Computational Linguistics. They consider two dimensions of each document: *relevance* and *rhetorical structure*. The first author manually annotated relevant (*i.e.*, salient) passages in all documents. As for the rhetorical structure, Teufel and Moens propose a non-hierarchical model of it. After manual examination of the corpus, they conclude that each paper in the collection has the following zones: AIM (the scientific research goal of the current paper), TEXTUAL (statements about the organization of the paper), OWN (description of the author's own work), BACKGROUND, CONTRAST (comparison with or contrast to other work), BASIS (statements of agreement with other work) and OTHER (description of other researchers' work). In order to

[8] PALSUMM is a proprietary system. For this reason, the amount of information available about its internal workings is lower than for some other systems which I reviewed in this dissertation.

determine the rhetorical category of a sentence and its relevance, each sentence is represented as a vector of features: location-related features, sentence length, presence of title words, verb-related features, presence of citations and presence of discourse markers. Teufel and Moens train several machine learners to identify relevant sentences and to assign them an appropriate category. The system selects salient sentences so that a summary always contains sentences from certain categories (AIM, CONTRAST and BASIS) and occasionally from other categories (OTHER, OWN, BACKGROUND).

To conclude, employing discourse-level information in text summarization has proven to be useful. However, despite many advances in the NLP technology, identifying the discourse structure of a document remains a very challenging task. The existing tools are few and unreliable, or proprietary and not readily available. In addition, they have not been thoroughly tested on texts other than articles.

In addition, information about the discourse structure of documents is very likely to be even more useful if it is combined with other information, for instance, information about distribution of keywords. However, this research direction remains largely unexplored.

2.6. Summarization of fiction

2.6.1. Overview

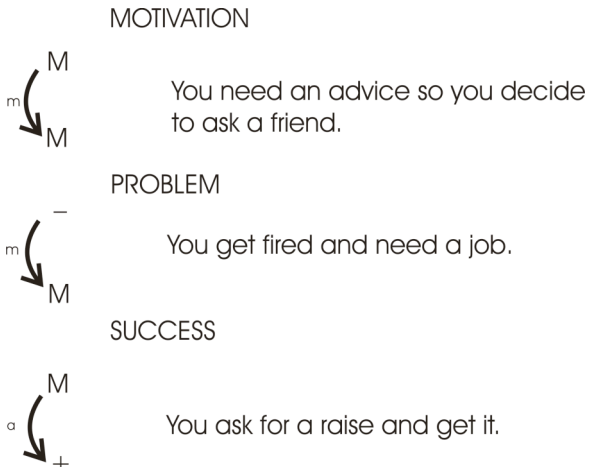
Most research initiatives of the text summarization community revolve around articles and scientific papers. However a handful of researchers attempted summarization-like projects of stories or fiction. The term “summarization of fiction” is perhaps a little inappropriate to describe this work; a better one would be “understanding of short stories or fiction”. This section contains an overview of several approaches in this direction.

2.6.2. Understanding fiction

Charniak (1972) made one of the earliest attempts to extract information from stories. Charniak works with children’s stories, short accounts of events that children can understand

effortlessly but that are a hard nut for a computer to crack (both in 1972 and today). The system accepts sentences that are represented using a particular language described by Charniak (1972). Sentence representation has three parts: the original text, a set of assertions that represent the sentence and a set of tags providing additional information about it. Both assertions and tags must be written using precise syntax and it is these parts of input that are used for further reasoning. Charniak's system creates a database of facts about the story, augmenting and correcting it as each new sentence becomes available. It attempts to keep the database non-redundant and updated. As a part of doing so, it makes inferences using the facts obtained from input and connects the facts using what Charniak calls DSP – deep semantic processing. Once a story has been input, a user can ask questions about it (again, using the specific input language).

Figure 2.5. Examples of primitive plot units (Lehnert 1982, p.380).



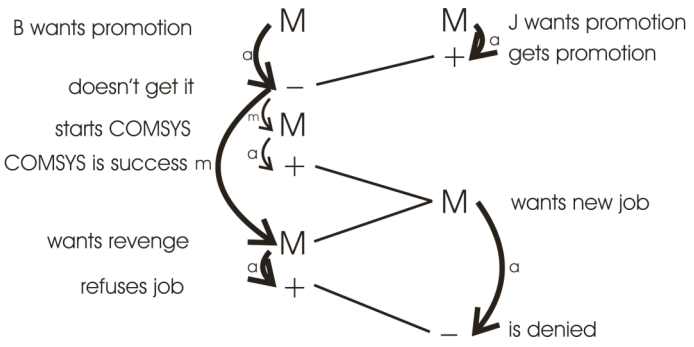
The obvious bottleneck of the system has to do with input: the program accepts only a very specific representation of sentences that is quite different from English. Therefore, it can only process stories for which such a representation is made available. On the other hand, the system has actually been implemented in LISP, which makes it more relevant to automatic text summarization (there never existed a concrete implementation of two other approaches that are reviewed in this section.)

Wendy Lehnert (1982) proposes an approach which relies on the idea that action-based stories (such as literary short stories) can be represented in terms of affect (or emotional) states of its characters. 'Emotional reactions and states of affect are central to the notion of a plot or a story structure' (Lehnert 1982, p.376.) She proposes a representation where a story can be decomposed into a number of *plot units*, which in turn consist of a number of atomic elements called *affect states*. There are only three possible affect states in (Lehnert 1982): + (a positive event), **-**(a negative event) or **M** (a mental state with neutral effect). A primitive plot unit consists

of affect states nodes connected by causal links, which can be of the following types: motivation (**m**), actualization (**a**), termination (**t**) and equivalence (**e**). A few examples of primitive plot units are offered in **Figure 2.5**. A more complex example of a complete representation of a story using plot units is shown in **Figure 2.6** (I refer to this story as *the COMSYS story* further in this review).

Figure 2.6. A complete plot-unit representation of a story (Lehnert 1982, p. 389)

John and Bill were competing for the same job promotion at IBM. John got the promotion and Bill decided to start his own consulting firm, COMSYS. Within three years COMSYS was flourishing. By that time John had become dissatisfied with IBM so he asked Bill for a job. Bill spitefully turned him down.



When comparing summaries of human subjects for the COMSYS story from **Figure 2.5**, Lehnert found that certain plot units, which she calls *pivotal units*, are more central to the story than others. In other words, the goodness of a summary is proportional to the number of pivotal plot units it includes. An ideal summary would include all pivotal plot units. (Lehnert 1982) proposes an algorithm for identifying pivotal plot units automatically, provided that a plot-unit representation of a story is available. Yet, it should be obvious to the reader that the construction of a graph of connected plot units for a story is far from trivial even for a human, let alone if it is to be done automatically.

Livia Polanyi (1989) takes yet another stance. Polanyi (1989) deals with stories as they may be told by one person to another[9]. According to Polanyi (1989), a story can be tentatively divided into parts that describe its background (*durative-descriptive clauses*) and parts that describe what actually happens (*event clauses*). She proceeds with an assumption that an author of a story is responsible for emphasizing important points and that she does so using a number of linguistic

[9] In fact, Polanyi specifies what she means by *person* very precisely. She makes a claim that delimiting historical, social and cultural aspects of story telling is crucial, because these aspects may alter not only understanding of a story, but what the term *story* means. For these reasons, she restricts her work to the stories such as may be told by white upper-middle class Americans.

devices such as elaboration in later clauses on information presented earlier, generalizations from an instance to a general case, flash sequences which provide explanatory information or reported speech (Polanyi 1989, p.24). Polanyi calls this process of emphasizing *evaluation* (not to be confused with the evaluation of results). (Polanyi 1989, p.27) proposes the following procedure for paraphrasing a story:

1. Divide the story into individual clauses or independent utterances.
2. Identify main event clauses and durative-descriptive clauses.
3. Prepare a list of the corresponding propositions.
4. Analyze the functioning of the evaluative meta-structure.
5. Calculate (at least roughly) the amount of evaluation accorded to each story-world proposition.
6. Combine the most heavily evaluated story events and durative-descriptive proposition into a stylistically acceptable paraphrase

2.6.3. Conclusion

As the reader might have gathered from this section, although there have been a few attempts to summarize fiction-like documents, these were neither numerous nor continuous. Therefore, although this dissertation borrows some abstract ideas from (Lehnert 1982) and (Polanyi 1989), it is mostly an exploration of an unknown territory.

2.7. Evaluation of text summarization

2.7.1. Overview

The discussion of automatic text summarization as a research area would not be complete without looking into the issues involved in evaluating the results. This is especially so because evaluating the goodness or the appropriateness of summaries is an immensely complex task. Making a judgment about what summary is better than others involves subjective judgment, depends on the context and on the purpose for which the summary is intended. In fact, if we had a

clear idea of what makes a good summary good, the task of automatic summarization would be much simplified. This section provides a brief overview of the issues involved in evaluating automatically produced summaries. I discuss challenges, a set of accepted practices and also a few recently proposed methods of evaluation. This section only discusses the evaluation, as it is relevant to summarizing structured documents because I am not aware of any research initiatives into the evaluation of summaries of literary works.

2.7.2. Intrinsic evaluation

The most commonly accepted practice of evaluating automatically created summaries is comparing them against gold-standard summaries (one or more). As a rule, a gold-standard summary is a summary created by a human (or humans.) This sort of evaluation is referred to as *intrinsic evaluation* since it evaluates the goodness of a summarizer by itself[10]. The shortcoming of this method is that no single summary, even if produced by a professional summarizer, can be judged to be the best. In other words, many people can say in many ways what essentially means the same thing. For instance, sentences A and B may be interchangeable in terms of meaning. Yet a person who produced a gold-standard summary chose to include sentence A only. If an automatically produced summary contains sentence B and not A, this substitution will be judged as an error.

There exist several ways to account for these discrepancies when creating gold-standard summaries. When more than one man-made summary is available, a researcher has several options with regard to which sentences are summary-worthy. One option is to judge a sentence summary-worthy only if the majority of human summarizers included it in their summaries. Other possibilities include taking a union or an intersection of the sentences that people consider summary-worthy.

However, all approaches that combine several man-made summaries into a reference one are inherently problematic because people never exhibit perfect agreement as to what should be in-

[10] On the other hand, extrinsic evaluation (**Section 2.7.3**) evaluates summaries by judging their appropriateness for a particular task

cluded in the summary. The most common measurement of the inter-judge agreement is *kappa* (Cohen 1960; Siegel and Castellan 1988):

$$K = \frac{P(A) - P(E)}{1 - P(E)}$$

In the formula above, *P(A)* is actual agreement between human summarizers and *P(E)* is expected agreement that the summarizers would agree by chance.

Researchers in text summarization report different statistics on the issue of agreement. When Rath et al. (1961) asked six human subjects to select 20 most informative sentences from 10 articles, the subjects agreed on an average of 1.6 sentence per article (8% agreement). Salton et al. (1997) report that two human summarizers displayed only 46% agreement when asked to summarize 50 articles by selecting five most informative paragraphs. Yet other studies report very high agreement between human judges. Jing et al. (1998) asked five subjects to summarize 40 articles. The subjects exhibited 96% agreement at 10% compression ratio and 90% agreement at 20% compression ratio. (Jing et al. (1998) use a different metric to measure the agreement, called *percent agreement*. This metric measures how well a particular judge agrees with the majority opinion.) Marcu (1997) reports 71% agreement between 13 human subjects that were asked to summarize five articles from *Scientific American*.

Mani (2001, p. 228) also points out another problem with this approach: no matter how many reference summaries are used to create a reference summary there is always a possibility that the system will generate something quite different which is still of good quality. "In other words, the set of reference summaries will necessarily be incomplete. This is especially true of generated abstracts."(Mani, 2001, p. 228)

Despite all its shortcomings, the intrinsic evaluation using a reference summary remains a very popular method of evaluation[11]. It has advantages that no other method available today offers: once a reference summary is constructed, the system can be automatically evaluated as many times as needed without incurring additional costs. In addition, the results are easily interpretable and, to some degree, comparable.

[11] Among the approaches reviewed in this dissertation, it has been used by (Marcu 2000; Teufel and Moens 2002; Zhou and Hovy 2005).

The discussion of intrinsic evaluation would not be complete without mentioning ROUGE (Lin 2004), a package for automatic evaluation of summaries. Given a set of reference summaries, ROUGE allows quick and inexpensive evaluation of the automatically produced ones. It offers a significant advantage over other methods of intrinsic evaluation: the reference summaries do not need to be extracted, people may write the summaries in their own words. This is an important advantage since it is more natural for people to write a summary from scratch than to select *n* most important sentences.

Given a set of reference summaries and a candidate one, ROUGE produces a set of scores that measure the similarity between them. The similarity measurements include n-gram recall (ROUGE-N score), the longest common subsequence (ROUGE-L) and the number of common word pairs with allowance for arbitrary gaps (ROUGE-S).

ROUGE-N score measures the n-gram recall between a reference summary and a candidate one. It is computed according to the following formula:

$$ROUGE-N = \frac{\sum_{S \in \{reference\_summaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{reference\_summaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

In this formula, *n* is the length of an n-gram and $Count_{match}$ is the number of n-grams that both summaries share.

When more than one reference summary is available, ROUGE-N score is the highest of the pairwise scores between each reference summary (r_i) and the candidate one (s):

$$ROUGE-N_{multi} = \arg\max_i ROUGE-N(r_i s)$$

ROUGE-L is an F-measure that estimates the similarity between a reference summary X of length m and the candidate summary Y of length n. The basis of the measurement if finding the longest common substring (LCS):

$$\mathrm{Re}call_{LCS} = \frac{LCS(X,Y)}{m}$$

$$\mathrm{Pr}ecision_{LCS} = \frac{LCS(X,Y)}{n}$$

$$F_{LCS} = \frac{(1+\beta^2)\mathrm{Re}call_{LCS}\,\mathrm{Pr}ecision_{LCS}}{\mathrm{Re}call_{LCS} + \beta^2\,\mathrm{Pr}ecision_{LCS}}$$

Where *LCS(X,Y)* is the length of the longest common subsequence of summaries *X* and *Y* and $\beta = Precision_{LCS} / Recall_{LCS}$ when $\delta FLCS / \delta Recall_{LCS} = \delta F_{LCS} / \delta Precision\ LCS$. In practice, β is set to a very large number and, therefore, only $Precision_{LCS}$ is considered.

In order to compute summary-level ROUGE-L, (Lin 2004) uses the following formulae:

$$\mathrm{Re}call_{LCS} = \frac{\sum_{i=1}^{m} LCS \cup (r_i, C)}{m}$$

$$\mathrm{Pr}ecision_{LCS} = \frac{\sum_{i=1}^{u} LCS \cup (r_i, C)}{n}$$

$$F_{LCS} = \frac{(1+\beta^2)\mathrm{Re}call_{LCS}\,\mathrm{Pr}ecision_{LCS}}{\mathrm{Re}call_{LCS} + \beta^2\,\mathrm{Pr}ecision_{LCS}}$$

u is the number of sentences in the reference summary and $LCS \cup (r_i, C)$ is the LCS score of the union longest common subsequence between a reference summary r_i and a candidate summary *C*.

ROUGE-S measures the similarity of a pair of summaries based on how many *skip-bigrams* they have in common. A *skip-bigram* is any pair of words in a sentence, allowing for arbitrary gaps. For instance, a sentence *Police killed the gunman* contains the following skip-bigrams: *police killed, police the, police gunman, killed the, killed gunman, the gunman.* Given a reference summary *X* of length *m* and the candidate summary *Y* of length *n* , ROUGE-S is computed in the following manner:

$$\mathrm{Re}call_{skip2} = \frac{SKIP2(X,Y)}{C(m,2)}$$

$$\mathrm{Pr}ecision_{skip2} = \frac{SKIP2(X,Y)}{C(n,2)}$$

$$F_{skip2} = \frac{(1+\beta^2)\mathrm{Re}call_{skip2}\,\mathrm{Pr}ecision_{skip2}}{\mathrm{Re}call_{skip2} + \beta^2\,\mathrm{Pr}ecision_{skip2}}$$

SKIP2 (X,Y) is the number if skip-bigram matches between summaries *X* and *Y* and *C* is the combination function.

Ever since its introduction, ROUGE package has been extensively used[12]. It has an immense advantage of being simple and inexpensive to use and can, therefore, be employed efficiently within the development cycle. However, it has a number of shortcomings. The interpretation of all ROUGE scores is far from trivial: it is not quite clear how the number of n-gram or skip-bigram matches affects the quality of a summary. In addition, all ROUGE scores evaluate only one aspect of the candidate summaries: their lexical overlap with the reference ones. Other aspects - such as informativeness, coherence or linguistic quality – remain unevaluated.

2.7.3. Extrinsic Evaluation

Extrinsic evaluation involves evaluating summaries by measuring their suitability for performing a specific task.

Jing et al. (1998) evaluate three machine-made and one man-made summary for each document in their corpus by measuring their usefulness for an information retrieval task. The corpus included four collections of 10 documents each. Each collection was either relevant or irrelevant for a specific query. The authors asked 12 subjects to judge if a document was relevant to a query in three distinct experimental settings: the subjects had to pronounce their judgment either using an automatically produced summary, a man-made one or using a full text.

Mani and Bloedorn (1997) report a similar experiment. The experiment involved four subjects and four collections of 75 documents each. Each collection was retrieved using a specific query and SMART information retrieval system (Buckley et al. 1993). The subjects had to judge whether a document was relevant to a query in two settings: using a query and a summary or using a query and an original document.

It is likely that extrinsic evaluation is a better way to measure the informativeness of a summary than comparing it to a reference summary on per-sentence basis. However, it is rather complicated to set up and costly to re-run. Perhaps this is the reason why it has been more of an exception than a rule to the common practice.

[12] Among the works reviewed in this dissertation, ROUGE has been used by (Filatova and Hatzivassiloglou 2004; Mihalcea and Tarau 2005; Li et al. 2006)

2.7.4. Evaluating the semantic content of a summary

In recent years (2003 and 2004) more and more people began to point out the need for more sound techniques for evaluating automatically produced summaries. As a result, two novel methodologies have emerged. These methodologies emphasize evaluating the semantic content and not the surface overlap with a reference summary.

Figure 2.7. Example factoids (van Halteren and Teufel 2003).

Sentence:
The police have arrested a white Dutch man.

Factoids:
FP20 A suspect was arrested
FP21 The police did the arresting
FP24 The suspect is white
FP25 The suspect is Dutch

Van Halteren and Teufel (2003) propose using *factoids* to evaluate the information content of a summary. Factoids are semantic expressions that follow the style of first order predicate logic (FOPL). **Figure 2.7** shows an example of a sentence and its decomposition into five factoids. The authors develop this representation after conducting an experiment where 50 subjects wrote a summary of the same article. Each summary was annotated for the presence of factoids. Van Halteren and Teufel (2003) report that while a single summary contains between 32 and 55 factoids, the collection as a whole contains 256 distinct factoids. It is interesting to note that, upon examination, only 169 of these are correct – other 87 are 'creative' factoids, which contain information that the human summarizers inferred from their world knowledge, not from the article. The authors state that the total number of factoids obtained from summaries of a single document is not fixed. Yet, they find that the distribution of the number of factoids is near-Zipfian and tends towards a certain number given a large number of summaries. Van Halteren and Teufel (2003) propose to measure the goodness of a summary by measuring how many most frequent factoids it contains.

Nenkova and Passonneau (2004) propose another approach that measures the semantic content of summaries. *The pyramid approach* also relies on the availability of a certain number of man-made reference summaries, although the number is significantly smaller (six in (Nenkova and Passonneau 2004) *vs.* 50 or more in (van Halteren and Teufel 2003).) The method relies on annotating reference summaries for presence of *Summarization Content Units* (SCUs), clause-

like pseudo-semantic fragments of text. Nenkova and Passonneau (2004) do not provide a formal definition of a SCU. Instead, they offer a detailed manual for identifying SCUs in the summaries.

When the annotation is complete, the authors propose to construct an abstract representation of importance of SCUs in reference summaries. This representation is called a *pyramid*. For an experiment involving *n* summaries of a document, an *n*-tier pyramid is constructed. Level 1 of the pyramid includes the SCUs found in only one summary, level 2 – the SCUs found in two summaries, level *n* – the SCUs found in all *n* summaries.

In order for an automatically produced summary to be evaluated, it must also be annotated for presence of SCUs. The score of a summary depends on the number of SCUs it contains and also on their relative importance (the level in the pyramid).

Factoids and pyramids are promising and much needed advances in the area of evaluating automatically produced summaries. However, at the moment, both methods are rather labour-intensive: not only do they require a significant number of reference summaries, but both reference summaries and candidate summaries need to be annotated for the presence of pseudo-semantic units. In addition, as the experiments of van Halteren and Teufel (2003) show, the annotation process is not simple even for coherent man-made summaries. What would the case be for occasionally very incoherent machine-made ones? This precludes such methods from being used within a development cycle. Yet, they remain the only methods that explicitly evaluate the semantic content of a summary.

2.8. Conclusion

This concludes my overview of the challenges and approaches specific to automatic text summarization. As a reader might have understood, the research area as a whole is an empirical one and lacks a sound theoretical foundation (either linguistic or computational). Yet, despite all odds, it rapidly advances and produces an impressive number of distinct methodologies. Different systems rely on different strategies to find salient portions of the text. However, no system combines all of these approaches.

Most research initiatives in the text summarization community revolve around summarizing well-structured documents. Very little has been done in summarizing fiction, and most of these

cannot be easily made algorithmic. This fact positions this dissertation to be a pilot study more than a continuation of an already advanced research area.

Chapter 3. Overview of the Proposed Method

3.1. Chapter overview

This dissertation deals with a novel task of summarizing fiction, namely short stories. Therefore, it is necessary to define the exact nature of the task and to state the objective in precise terms. **Section 3.2** discusses briefly the nature of short stories in general and explains why I chose this and not any other genre of fiction for this work. **Section 3.3** contains the description of the corpus of short stories and of its characteristics. **Section 3.4** explains what I mean by the term *a summary of a short story* in this dissertation and delimits the scope of what I strove to achieve in the course of this work. **Section 3.5** provides a bird's-eye view of my approach to constructing such summaries and the motivation behind it.

3.2. On the nature of short stories

Despite the apparent clarity of the term *short story*, its formal definition is difficult, if not impossible, to come by. Allan Pasco defines short stories as "short, literary prose fiction" (Pasco 1991, p. 411.) He elaborates that in the common sense of the word this implies a written work that deserves a considerable degree of aesthetic merit (that is, it needs to be artistic), that introduces a world that cannot be verified externally (the setting and the plot) and that, according to Edgar Poe's classic definition, can be read in one sitting (Poe 1842, p. 61). This definition is not formal in the technical sense of the word, but one can hardly expect to find such a definition for a genre of art. Other definitions in accord with the one above can be found in (Poe 1842; Matthews 1888; Bates 1972).

Certain characteristics of short stories set them apart from genres such as novels and novelettes. According to Brander Matthews, a "short-story deals with a single character, a single event, a single emotion, or the series of emotions called forth by a single situation" (Matthews

1888, p. 73.) This statement emphasizes the most important characteristic of the short story as a genre: its marked tendency towards totality (Poe 1842), or unity of impression (Matthews 1888).

There is more agreement on the typical constituents of short stories. The majority of short stories have the following elements: 1) characters, 2) a setting, 3) a plot, 4) a conflict and 5) a theme (or moral) (Gervig 1971; Knight 1981). Depending on the time of writing and the author's background and intentions, some of these constituents may be emphasized more than the others: authors such as Guy de Maupassant and Anton Chekhov place more emphasis on the imagery and their stories tend to be more descriptive (Pasco 1991; May 1994). Other, such as O. Henry or Edgar Allan Poe, emphasize the plot (May 1994). Yet all but few exemplars of the genre exhibit all these elements in one form or another.

This dissertation focuses on the automatic summarization of fiction and I chose short stories as an example of the genre suitable for an initial exploration. There are several reasons for this decision. It was my intuition and it is backed by literary criticism that short stories are less complex structurally than, for instance, novels or novellas. As a rule, a short story revolves around a small number of central characters: one, two, seldom more (Matthews 1888). Unlike longer genres, short stories are *single-valent*: that is, they rarely have sub-plots, as, for instance, novels do (Pasco 1991). They exhibit a marked tendency towards unity and non-redundancy (Pasco 1991). These characteristics, coupled with the relatively small size of short stories, make this genre suitable for an initial exploration of this new type of data. As I already mentioned in **Chapter 1**, the brevity of short stories made it feasible to collect a new corpus, manually annotate it and evaluate the results in a meaningful way. This might not have been the case had longer works been involved.

3.3. Corpus description

In the course of this project I worked with a small corpus that consisted of 47 short stories collected from Project Guttenberg (http://www.gutenberg.org). I manually selected the stories before starting any experiments and before making any decisions about the approach.

Several criteria dictated which stories to include in the corpus. First of all, since this work is an exploration of automatic summarization of a new type of data, the stories had to be typical

examples of the genre. Therefore, using classics of the genre seemed to be a natural choice. Another desired characteristic of the corpus was that the stories be written in English similar to English we use today (that is, short stories written during the XV century would not quite do). This was especially important because most Natural Language Processing tools have been created for contemporary, not old English. Yet, an opposite constraint also needed to be considered: the issue of copyright. Because of the copyright restrictions, the corpus only contains stories that are in the public domain due to copyright expiration. In addition, given the exploratory nature of the project, I took care not to include stories that are overly complex. The meaning of complexity is two-fold here: linguistic and psychological. The linguistic complexity as used here refers to the complexity of discourse in short stories: I tried not to include stories characterized by the excessive amount of dialogue and stories where the chronological order of events was severely different from linear, such as flashbacks common to the stream-of-consciousness style of writing. The second type of complexity is psychological: occasionally, some stories cannot be retold, let alone summarized. I tried not to include such stories in the corpus.

Because of these reasons, the final corpus consists of 47 short stories written by the mainstream XIX-XX century authors such as O.Henry, Jerome K. Jerome, Katherine Mansfield and Anton Chekhov[13]. A complete list of the stories used in the corpus appears in **Appendix A**. An average length of a story in the corpus is 3,333 tokens (approximately 4.5 letter-sized pages).

3.4. Defining the objective

Now that the nature of the data used has been defined, it is useful to give a clear-cut definition of what this work intended to achieve.

When starting this work, I defined my objective as follows: *to produce extractive summaries of short stories in such a way that they help a reader decide whether she would be interested in reading a particular story without revealing the plot*. This definition combines several statements and, therefore, needs clarification.

First of all, let us consider the term *extractive*: this means that the summaries are to be composed of sentences extracted from the original without any modifications. This choice is due to

[13] Some of the stories are translations into English.

the novelty and complexity of the task. Constructing an abstract as opposed to an extract would add yet another layer of complexity to this already difficult problem. Such an attempt seemed premature. Therefore, the task is to identify important sentences in original stories[14].

The objective also states that the summaries should help a reader decide whether she would be interested in reading a particular story without revealing the plot. In other words, after reading a summary, a reader should have adequate expectations as to what sort of story is to come and whether she would like to read it. There are two reasons for excluding the plot from the summary. The first reason is practical: most people do not like knowing the plot beforehand, even if it helps them decide to read or not to read the story. The second reason for excluding the plot from summaries is empirical: identifying important events of the plot would be considerably more challenging than finding enough information to raise adequate expectations about a story. This task constitutes a direction in which this work may further evolve.

3.5. Overview of the Approach

As I have said in **Section 3.2**, the majority of short stories have the following constituents (Gervig 1971; Knight 1981):

1. The setting.
2. The characters.
3. The plot.
4. The conflict.
5. The theme (or moral).

[14] This decision is in line with the state-of-the-art within the text summarization community: only very few researchers (Barzilay and McKeown 2005) have ventured into going beyond extraction-based summaries.

Figure 3.1. An example of a manually created summary.

The Cost of Kindness.
Jerome K. Jerome (1859-1927).

The Rev. Augustus Cracklethorpe would be quitting Wychwood-on-the-Heath the following Monday, never to set foot--so the Rev. Augustus Cracklethorpe himself and every single member of his congregation hoped sincerely--in the neighbourhood again. Hitherto no pains had been taken on either side to disguise the mutual joy with which the parting was looked forward to. The Rev. Augustus Cracklethorpe, M.A., might possibly have been of service to his Church in, say, some East-end parish of unsavoury reputation, some mission station far advanced amid the hordes of heathendom. There his inborn instinct of antagonism to everybody and everything surrounding him, his unconquerable disregard for other people's views and feelings, his inspired conviction that everybody but himself was bound to be always wrong about everything, combined with determination to act and speak fearlessly in such belief, might have found their uses. In picturesque little Wychwood-on-the-Heath, among the Kentish hills, retreat beloved of the retired tradesman, the spinster of moderate means, the reformed Bohemian developing latent instincts towards respectability, these qualities made only for scandal and disunion.

Matters had come to a head by the determination officially announced to him that, failing other alternatives, a deputation of his leading parishioners would wait upon his bishop. This it was that had brought it home to the Rev. Augustus Cracklethorpe that, as the spiritual guide and comforter of Wychwood-on-the-Heath, he had proved a failure.

Churchgoers who had not visited St. Jude's for months had promised themselves the luxury of feeling they were listening to the Rev. Augustus Cracklethorpe for the last time.

What marred the entire business was the impulsiveness of little Mrs. Pennycoop.

This dissertation concentrates on identifying salient elements of the first two constituents: the setting and the character descriptions. The goal is to select sentences that tell whom the story is about and where and when it takes place. This information should be organized into a relatively coherent summary[15] that goes beyond simply listing names and places. This choice is based on an assumption that once a reader has a summary that provides her with salient information about the setting of a story and its central characters in the language of the original, she will able to form adequate expectations as to what is to come and to make informed decisions about the complete story.

[15] By relatively coherent I mean as coherent as an extractive summary can be, as there are inherent limitations to that. A summary that is composed of sentences extracted from various parts of the original cannot be as coherent as a summary that was manually written and has a certain level of abstraction.

An example of a summary created following these guidelines for *The Cost of Kindness* by Jerome K. Jerome is shown in **Figure 3.1**[16]. The summary gives the reader enough information to understand that the story takes place in a small town in the English countryside and that it is about a generally disliked vicar who is leaving the town. The reader also knows that something "marred the entire business." The plot, which relates how one local family decides to bid a warm farewell to the Rev. Cracklethorpe, the whole town follows their example and it causes the vicar to change his mind and stay, is not included in the summary. It is up to a reader to discover, should he decide to do so.

The system works in two stages: 1) identifying salient entities in sentences (such as characters, locations and temporal anchors) and 2) selecting descriptive sentences (as opposed to event sentences)[17]. By doing so I expect to produce summaries that achieve the stated objective: they raise adequate expectations about the original and help a reader decide whether she wants to read the story.

In order to identify important entities in sentences, the corpus is processed using a gazetteer that identifies locations, dates and animate entities. In addition, anaphoric references to animate entities (*i.e.*, characters) are resolved. **Chapter 4** describes this preparatory stage of the project.

Chapters 5 and **6** explain the actual sentence selection process. This process relies heavily on the notion of the grammatical aspect. The system computes a number of indicators that help determine the grammatical aspect of every clause and selects descriptive rather than event clauses. **Chapter 5** provides the linguistic motivation for this decision and explains the notion of the grammatical aspect and what indicators signal it. **Chapter 6** describes the sentence selection procedures.

[16] This summary was created by one of the annotators according to the procedure described in **Chapter 7**.
[17] By *descriptive* sentences I mean sentences that relate the background of a story. The *event* sentences talk about events, or what happens in the story. These definitions are in line with the definitions of *durative-descriptive clauses* and *event clauses* from (Polanyi 1989).

Chapter 4. Identification of Important Entities in Short Stories.

4.1. Chapter overview

This chapter describes the preparatory stage of the project. The purpose of this stage is to identify salient entities in short stories. These entities are main characters, locations and dates.

In order to find such entities, the corpus is processed using GATE software (Cunningham et al. 2002) that identifies locations, dates and animate entities. However, GATE (as well as other named-entity recognizing software) does not recognize all mentions of animate entities: it does not, for instance, recognize anaphoric expressions, which are very common in fiction. In order to make this information available, I implemented an anaphora-resolution module.

Section 4.2 describes the process of recognizing animate entities, locations and dates using GATE. **Section 4.3** discusses the phenomenon of anaphora and describes the anaphora resolution module, which is a part of this system. It also reports the results of a small-scale evaluation of the module. **Section 4.4** explains how the system distinguishes between important and unimportant characters in each story. **Section 4.5** concludes the description of this part of the system.

Section 4.6 is a review of a few alternative approaches to anaphora resolution. It also explains why I chose the algorithm from **Section 4.3** and not any other.

4.2. Identifying salient entities

The system for summarizing short stories consists of two main parts: a module that identifies important entities (*e.g.*, characters, location and dates) and a sentence selection module.

In order to identify salient entities, the corpus is processed using GATE (Cunningham et al. 2002). GATE is an open-source framework for developing Natural Language Processing applications. It contains a variety of ready-to-use tools and, among others, a gazetteer and a named-entity recognizer. GATE recognizes dates, locations and several kinds of expressions denoting

people, such as simple and complex personal names (*e.g.,* both *John* or *M. Jacques Chirac* would be recognized*)* and job titles (*e.g., a secretary*, *a professor*, *etc*.).

In order to clarify what types of entities (people, places or dates) are most frequent within the corpus, I conducted a small experiment with 14 stories from the training set. Each story in this subset was processed using GATE and I manually counted the number of animate entities, location and dates that GATE identified. The findings are as follows: each story contained multiple mentions of characters (on average, 64 mentions per story). Yet I found only 22 location markers, most of these street names. The 22 markers were found in 10 out of 14 stories, leaving 4 stories without any identifiable location markers. Only 4 temporal anchors were identified in all 14 stories: 2 absolute (such as years) and 2 relative (names of holidays). These findings support the intuitive idea that short stories revolve around their characters, even if the ultimate goal is to show a larger social phenomenon. The idea is also echoed in (Lehnert 1982; Knight 1981). The findings also suggest that looking for time stamps in short stories is unlikely to prove productive, as such information is not included in these texts explicitly. That is why my system does not attempt to identify them.

Since character mentions appear to be frequent in fiction, I designed the pre-processing stage of the project so as to maximize the amount of character-related information that it made available to subsequent stages. I extended the GATE gazetteer to recognize common nouns denoting animate entities, not only persons' names and professions. As the gazetteer is list-based, this amounted to including commonly used nouns that denote people and animals in separate lists. The lists of nouns that I added to the gazetteer appear in **Appendix B**. Subsequently, I created an anaphora resolution module that resolves 1st and 3rd person singular pronouns and singular nominal references to animate entities (characters in the story). The resolution of the anaphoric expressions ensures that the character-related information is used with maximum efficiency. The following section explains the concept of anaphora and describes the anaphora-resolution module implemented within this system.

4.3. The anaphora resolution module

4.3.1. The concept of anaphora

The term *anaphora* can be explained as a way of mentioning a previously encountered entity without naming it explicitly. Consider *Examples 4.1* and *4.2* from *The Gift of the Magi* by O. Henry. *4.1* is an example of pronominal anaphora, where the *noun phrase* (further *NP*) *Della* is referred to as an *antecedent* and both occurrences of the pronoun *her* as *anaphoric expressions* or *referents*. *Example 4.2* illustrates the concept of noun phrase anaphora. Here the NP *Dell* is the antecedent and *my girl* is the anaphoric expression (in the context of this story *Della* and the *girl* are the same person).

(Example 4.1) Della$_{i1}$ *finished* her $_{i1}$ *cry and attended to* her $_{i1}$ *cheeks with the powder rag.*

(Example 4.2) *"Don't make any mistake,* Dell $_{i1}$*," he said, "about me. I don't think there's anything […] that could make me like* my girl $_{i1}$ *any less.*

Such expressions are very common in speech and written text; this is especially true about short stories where characters play an important role. Resolving these expressions (that is, making available the fact that *Della* and *her* refer to the same person) would significantly increase the amount of character-related information. [18]

The primary goal of the anaphora resolution module in this system is increasing the amount of information about main characters. This is why the resolution of anaphoric expressions was limited to resolving noun phrase and pronominal anaphora denoting animate entities: mostly people and some animals. The anaphora resolution module handles the following types of anaphora:

- 3rd and 1st person singular pronouns denoting animate entities[19]
- singular definite noun phrases denoting animate entities (e.g., *the man*, *that student*).

18 Pronominal and noun phrase anaphora are not the only possible types of anaphora: there exist also verbal anaphora and ellipsis, but handling these is beyond the scope of this work.

19 I decided against resolving other pronouns because the resolution of the 2nd person pronoun (*you)* requires discourse-level knowledge and resolving the pronoun *it* requires semantic knowledge in order to perform the resolution reliably. Resolving plural pronouns (*e.g.*, *we, you, they)* is also complicated because no gender information is available in these cases and because the system needs to find more than one antecedent. Gender information is also unavailable for the pronoun *I* , but I rely on a heuristic to resolve such pronouns (see **Section 4.3.3**).

The module consists of two main parts: the part responsible for the resolution of pronominal anaphora and the part resolving noun phrase anaphora. The pronoun resolution algorithm is an implementation of the very classical algorithm proposed by Lappin and Leass (1994). **Section 4.3.2** describes the details of the algorithm as implemented here. The noun phrase anaphora resolution sub-module combines the approach to identifying anaphoric noun phrases proposed by Poesio and Vieira (2000) with the algorithm of Lappin and Leass (1994) adapted to finding antecedents of definite NPs. It is described in **Section 4.3.4**.

It should be clear to the reader that the anaphora resolution module is an implementation of the algorithms proposed by other researcher. I provide the details of both algorithms here for completeness. However, a reader not interested in the details of the implementation is encouraged to proceed to **Section 4.3.6** (**Evaluation of the anaphora resolution module**).

4.3.2. Tools used in the anaphora resolution module

The anaphora resolution module is an implementation of a syntactically oriented algorithm proposed by Lappin and Leass (1994). This algorithm relies on the output of a syntactic parser; the original version was implemented using the English Slot Grammar parser (ESG parser) (McCord 1980). The implementation within this system is adapted to use the output of the Connexor Machinese Syntax Parser (further the *Connexor parser*), version 3.8 (Tapanainen and Järvinen 1997).

Figure 4.1. An example of the Connexor parser output.
Sentence text: Della finished her cry.

C1	C2	C3	C4	C5
1	Della	della	subj:>2	@SUBJ %NH N NOM SG
2	finished	finish	main:>0	@+FMAINV %VA V PAST
3	her	she	attr:>4	@A> %>N PRON PERS GEN SG3
4	cry	cry	obj:>2	@OBJ %NH N NOM SG
5	.	.		
6	<s>	<s>		

An example of the Connexor parser output for the sentence *Della finished her cry* appears in **Figure 4.1**. The parser provides several types of information about each token within a sentence. Column *C1* assigns a unique id to each token in the sentence and column *C2* contains the token itself. Column *C3* displays the base form, or the *lemma*, (*e.g.*, *finish* for the token *finished*). Column *C4* shows the dependency relation with the parent node in the parse tree and the id of the

parent node (for instance, the token *Della* is linked by a dependency of type *subject* to its parent node *2*, *finished*). Column *C5* is a concatenation of several fields: syntactic function tags (beginning with @), surface syntactic tags (beginning with %) and morphological tags (the rest of information). For instance, one can see the following information for the token *Della* with id *1*: the node's syntactic function is subject (@SUBJ), it is a head of a nominal construction (%NH), it is a noun (N), used in the nominative case (NOM) and in singular form (SG).

Figure 4.2. An example of a JAPE rule.

```
Rule: PersonNoun

 (person_noun)
 (person_name) +
: person
```

This rule would recognize such expressions as *aunt Mary* or *doctor John Smith* (assuming that the nouns *aunt* and *doctor* can be recognized as *person_noun*).

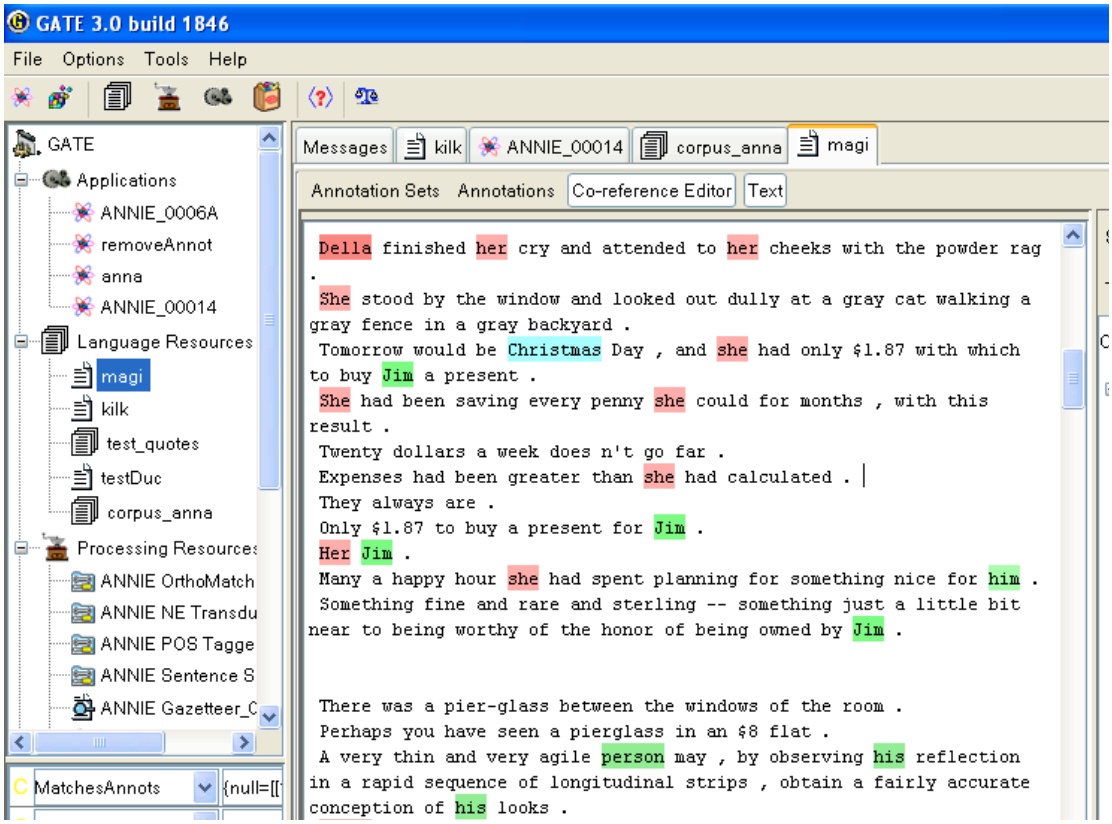
The output in **Figure 4.1** is in textual format, but the parser also produces the output in XML. In this work, the latter option was used. The resulting XML files containing parse trees are transformed[20] so as to allow them to be loaded into GATE. The GATE framework (Cunningham et al. 2002) is an effort towards creating a unified platform for Natural Language Engineering. It offers a wealth of ready-to-use tools, such as part-of-speech taggers, machine learning tools, various ontologies and information retrieval systems[21]. GATE allows combining these resources in a seamless manner by enforcing input/output standards and interfaces between its plug-ins. GATE is implemented in Java and its main strength is that it allows the creation and integration of new resources and plug-ins without expending much energy on interfaces. The anaphora resolution module described in this section is implemented in Java as a GATE plug-in (processing resource).

I used GATE for two reasons. GATE gazetteer and named-entity recognizer used in combination recognize people, locations, dates and several other types of entities. Despite the fact that the gazetteer is list-based, the GATE internal language, JAPE, allows creating regular expressions over gazetteer annotations. This makes it possible to recognize rather complex expressions. **Figure 4.2** shows an example of a very simple JAPE regular expression. The gazetteer annotates expressions denoting animate entities with gender information: *Mrs. Smith* would be

[20] The transformation is XML to XML. Its only purpose is to produce GATE-compatible XML files.

[21] For a complete list of the available tools see the list at http://gate.ac.uk/gate/doc/plugins.html

Figure 4.3. An example of an anaphora-resolved document viewed in GATE.



marked as female and *Mr. Smith* - as male. This information is invaluable for the anaphora resolution module because it allows performing gender agreement checks between a possible antecedent and a referring expression.

The second reason for using GATE as a framework for the anaphora resolution module is its graphical user interface. Without any effort on the part of a developer of new plug-ins, GATE offers an easy way to visualize the output. This capability made debugging and fine-tuning of the module considerably easier. **Figure 4.3** shows an example of a document with resolved anaphoric references.

4.3.3. Resolution of pronominal anaphora

The pronoun resolution sub-module proceeds in the following manner. An XML document containing a parsed short story is loaded into GATE. GATE identifies dates, locations and

Figure 4.4.a. An implementation of the pronoun resolution algorithm of Lappin and Leass (1994) (continued in Figure 4.4.b).

Input: *allCandidates*, *windowSize*, *sentences*, *Quotes*, *recencyDecayFactor*

```
0 def  Execute:
1 for sentence in sentences do
2         curTree = construct a parse tree for sentence
3         //collect possible antecedents from this sentence and add them to the common list
4         ProcessSentCandidates ( curTree )
5         pronouns = collect pronouns from sentence
6         if pronouns == null
7                 UpdateCandidates (allCandidates, sentence)
8                 continue
9         //otherwise we have some pronouns to resolve
10        for pronoun in pronouns do
11                if pronoun is 1st person pronoun // e.g. I, me, my, etc.
12                        antecedent = ResolveFirstPerson (pronoun, sentence)
13                else if pronoun is 3rd person pronoun //e.g. she, her, he, etc.
14                        antecedent = FindAntecedent ( pronoun, allCandidates )
15                        if antecedent ==  null
16                                //this means that no antecedent was found in the preceding
17                                //sentences. Perhaps it can be found in the succeeding sentences
18                                cataphoraCandidates = collect possible antecedents from up to
19                                                        windowSize succeeding sentences
20                                antecedent = FindAntecedent ( pronoun, cataphoraCandidates )
21                if antecedent != null
22                        link pronoun to antecedent
23                UpdateCandidates ( allCandidates, sentence )
24
25 def ProcessSentCandidates (sentenceTree):
26        collect all NPs denoting animate entities within sentenceTree
27        remove NPs in plural
28        rank NPs according to their syntactic function and recency
29
30 def UpdateCandidates ( candidates, sentence ):
31        curSent = index of the current sentence being processed
32        for candidate in candidates do
33                candSentIndex = index of a sentence where this candidate was encountered
34                distance = candSentIndex – curSent
35                if distance < windowSize
36                        //penalize older sentences
37                        candidate.score = candidate.score * recencyDecayFactor
38                else      //this candidate is too old
39                        remove candidate from candidates
40
```

continued in **Figure 4.4.b**.

animate entities; it also annotates the latter with gender information. Then the program processes sentences one by one (lines 1-23 of pseudo-code in **Figure 4.4.a**). A parse tree corresponding to the sentence is loaded into memory (using the information supplied by the Connexor parser) (line

Figure 4.4.b. An implementation of the pronoun resolution algorithm of Lappin and Leass (1994) (continued from Figure 4.4.a).

```
41 def FindAntecedent ( candidates, refExpression ):
42         possibleAntecedents  = null
43         for candidate in candidates do
44                 if AgreementCheck ( refExpression, candidate ) == True
45                         if SyntacticCheck ( refExpression, candidate ) == True
46                                 possibleAntecedents.append ( candidate )
47         antecedent = select highest ranking antecedent from possibleAntecedents
48         curScore = calculate syntactic score of refExpression
49         candidates [ antecedent ] . score = curScore
50         if antecedent.gender == null
51                 antecedent.gender = refExpression.gender
52         return antecedent
53
54 def ResolveFirstPerson ( pronoun, sentence ):
55          curQuote = find the quoted text span where this pronoun is encountered
56         if curQuote == null
57                 //this 1st person pronoun was encountered outside any quoted text spans
58                 return 'NARRATOR'
59         //otherwise this pronoun was found inside quotes
60          candidates = collect candidates from up to windowSize sentences before the opening quote
61          antecedent = FindAntecedent ( candidates, pronoun )
62          if antecedent == null
63                 candidates = collect candidates from up to windowSize sentences after the closing quote
64                 antecedent = FindAntecedent ( candidates, pronoun )
65          if antecedent == null
66                 candidates = collect candidates from up to windowSize sentences inside the quoted text
67                 antecedent = FindAntecedent ( candidates, pronoun )
68         return antecedent
```

2). Next, the program collects all entities that can potentially be antecedents of anaphoric expressions (in this work, these are singular NPs denoting animate entities). These entities are added to the global pool of possible antecedents (a call to the procedure *ProcessSentCandidates*, line 4). Each entity in the pool is ranked on the basis of its syntactic function and recency (line 28). Once a ranked list of possible antecedents has been created, the program performs two checks between a referring expression and each possible antecedent: a syntactic check (*SyntacticCheck* on line 44) and a check of number and gender agreement (*AgreementCheck* on line 44). In this manner some of the possibilities are eliminated. The highest-ranking antecedent is selected among the remaining candidates and the referring expression is linked to it.

Collecting and ranking candidate antecedents. The procedure *ProcessSentCandidates* (line 25) is responsible for collecting and ranking candidate antecedents based on their syntactic

function and recency. All factors that are taken into consideration when ranking candidate antecedents are listed in **Table 4.1**[22]. A rank of a candidate antecedent is the sum of weights of all factors that apply.

(Example 4.3) John plays basketball.

Let us consider the NP *John* from the sentence in *Example 4.3*. This noun phrase is the subject of a sentence, it contains a head noun (*John*), it is not embedded in an adverbial construction and it is the latest sentence we encountered. Therefore, the score of this NP is a sum of the following salience weights: subject, head noun, non-adverbial emphasis and recency. Its score is: 100 + 80 + 80 + 50 = 310.

Finding antecedents of pronouns. Once all potential antecedents from a sentence have been added to the common pool (line 4), the program attempts to resolve any pronouns that are found in that sentence.

The resolution procedure for both the 3rd and the 1st person pronouns is essentially the same, **FindAntecedent** on line 41. Yet there is a difference in the way the 1st person pronouns (*I*, *me*, *mine*, etc.) are handled. These pronouns are only encountered in two cases: in narratives written in 1st person or in dialogues. In dialogues, they appear as a part of direct speech, which is usually a span of quoted text. GATE can identify spans of quoted text[23]. This information is computed before the execution of the resolution module and is stored in a data structure *Quotes*. When a 1st person pronoun is encountered, the program checks whether the pronoun was found inside a span of quoted text. If this is not

Table 4.1. Salience weighting factors used for ranking candidate antecedents.

Factor type	Weight
Sentence recency (awarded to antecedents in the most recent sentence)	100
Subject emphasis e.g. *John plays basketball.*	80
Existential emphasis e.g. *There was a man.*	70
Accusative emphasis e.g. *Someone pushed the boy.*	50
Indirect object emphasis e.g. *Someone gave a book to the boy.*	40
Head noun emphasis *The father of the kid plays basketball.*	80
Non-adverbial emphasis (Nouns non embedded in adverbial constructions) *The boy, whose father played basketball, rose.*	50

[22] Lappin and Leass call these factors *salience factors*.

[23] This capability was subject to some serious changes and bug-fixing. The original facility provided by GATE was too error-prone.

true, the program assumes that it encountered a case of a 1st person narrative and returns a dummy antecedent *NARRATOR* (line 58). Otherwise (for pronouns found inside a span of quoted text), the corresponding quoted text object is identified (line 55). In order to find an antecedent of such 1st person pronouns, the following procedure is followed (lines 61-67): the program first looks for antecedents in the span of text preceding the quoted text (line 61). If no antecedent is found in this span, the second place to look is the span of text following the quoted text. If this too fails, then the program searches for an antecedent inside the quoted text. This way to handle 1st person singular pronouns was first proposed by Dimitrov (2002).

The syntactic filter. The procedure *FindAntecedent* performs the actual antecedent - pronoun matching[24]. It selects only those candidate antecedents that satisfy two constraints: agree with the pronoun in gender and number (line 44) and satisfy several syntactic constraints. The syntactic filter is only applied to pronoun - antecedent pairs found within the same sentence. In order to explain its inner workings, it is necessary to define a few terms (these definitions appear in (Lappin and Leass 1994)).

Let *F* and *G* be phrases from a parse tree *T*.

Argument domain: *F* is in *the argument domain* of *G* iff both *F* and *G* are arguments of the same head.

Adjunct domain: *F* is in the *adjunct domain* of *G* iff *G* is an argument of a head *H*, *F* is the object of a prepositional phrase *PREP* and *PREP* is an adjunct of *H*.

Noun phrase domain: *F* is in *the noun phrase domain* of *G* iff *G* is the determiner of a noun *Q*, and 1) *F* is an argument of *Q* or 2) *F* is the object of a prepositional phrase *PREP* and *PREP* is an adjunct of *Q*.

Containment: a phrase *F* is contained in a phrase *G* iff 1) *F* is either an argument or an adjunct of *G* or 2) *F* is an argument or an adjunct of some phrase *Q* and *Q* is either an argument or an adjunct of *G*.

It must be noted, that unlike the ESG parser, for which the algorithm was originally created, the Connexor parser does not differentiate between arguments and adjuncts of syntactic heads. For this reason, in this implementation, the terms *argument* and *adjunct* are synonymous with the term *immediate child*.

---

[24] As the reader will see, the same procedure is also resolves noun phrase anaphora.

Armed with these definitions we may at last describe the syntactic filter. A non-reflexive pronoun *P* cannot be co-referential with an antecedent *NP* if one of the following conditions holds:

1. *P* is in the argument domain of *NP*.
   John asked him.
2. *P* is in the adjunct domain of *NP*.
   She sat near her.
3. *P* is an argument of a head *H*, *NP* is not a pronoun and *NP* is contained in *H*.
   He believes that the man is amusing.
4. *P* is in the noun phrase domain of *NP*.
   John's portrait of him is interesting.
5. *P* is a determiner of a noun *Q*, and *NP* is contained in *Q*.
   His portrait of John is interesting.

The syntactic filter works differently for reflexive pronouns (e.g. *himself*). A reflexive pronoun *P* can be co-referential with the antecedent *NP* iff one of the following conditions holds:

1. *P* is in the argument domain of *NP* and *NP* fills a higher argument slot than *P*.
   They wanted to see themselves.
2. *P* is in the adjunct domain of *NP*.
   He worked by himself.
3. *P* is in the noun phrase domain of *NP*
   Mary likes Bill's portrait of himself.
4. *NP* is an argument of a verb *V*, there is a noun phrase *Q* in the argument domain or the adjunct domain of *NP* such that *Q* has no determiner and 1) *P* is an argument of *Q* or 2) *P* is an argument of a prepositional phrase *PREP* and *PREP* is an adjunct of *Q*.
   They told stories about themselves.
5. *P* is a determiner of a noun *Q* and 1) *Q* is in the argument domain of *NP* and *NP* fills a higher slot than *Q* or 2) *Q* is in the adjunct domain *NP*.
   They liked each other's portraits.

The loop on lines 43-46 in **Figure 4.4.b** identifies candidate antecedents that satisfy both conditions. The one with the highest score is selected as an antecedent for a referring expression

(line 47). The weight of the selected antecedent is boosted to reflect a recent mention (line 48-49): its new score equals the score of a referring expression calculated according to weighing scheme in **Table 4.1**. The selected antecedent is returned.

If no antecedent is found in the preceding sentences, the program checks whether the referring expression is *cataphoric*: that is, the referring expression precedes its antecedent (lines 15-20). In this case, the search for an antecedent is performed in an adjacent window of sentences that follow the sentence being inspected. This is only done for 3rd person pronouns.

Adjusting candidate antecedent weights. In order to reward recent candidates and to penalize older ones, the procedure *UpdateCandidates* (line 30 in **Figure 4.4.a**) adjusts the scores of candidate antecedents as they become less recent. It also removes the candidates that are more than *windowSize* sentences behind or ahead of a sentence being processed. Once the sentence is processed (line 23), the score for each candidate antecedent is multiplied by *decayFactor* (line 37), which was empirically set at 0.8 in this implementation[25]. Only the candidates from a window of fixed size are preserved in the list. The window size was set at 6 (the global variable *windowSize*)[26]. Once a candidate is more than *windowSize* sentences old, it is removed from the list of possible antecedents (line 39).

4.3.4. Resolution of noun phrase anaphora

The *noun-phrase anaphora* is a type of anaphora such that both the antecedent and the referring expressions are noun phrases (Mitkov 2002, p.10.) They may be realized as a definite noun phrase or as a proper name. This type of anaphora is known as *definite descriptions* (Poesio and Vieira 2000; Mitkov 2002, p.10.)

Not all definite NPs are anaphoric: *Example 4.4* shows a case when two definite NPs indeed co-refer, but *Example 4.5* shows a counter-example. The NP *(Old Antony Rockwall)* introduces a new entity (the sentence used in *Example 4.5* is the first sentence of *The Mammon and The Archer* by O. Henry.)

---

[25] The decay factor used in (Lapin and Leass 1994) has a different value, 0.5. But in their case, the algorithm was fine-tuned using a corpus of computer manuals.

[26] I fined-tuned both of these values using five stories from the training set.

Example 4.4 *John Armstrong*$_{i1}$ and *Mlle. Giraud*$_{i2}$ rode among the Andean peaks, enveloped in their greatness and sublimity. […] To *Armstrong*$_{i1}$ *the woman*$_{i2}$ seemed almost a holy thing. (O. Henry, *The Matter of Mean Elevation*).

Example 4.5 Old Anthony Rockwall$_{i3}$, retired manufacturer and proprietor of Rockwall's Eureka Soap, looked out the library window of his Fifth Avenue mansion and grinned. (O.Henry, *The Mammon and the Archer.*)

If a definite NP is anaphoric, it may be a case of *direct anaphora* or *indirect anaphora. Direct anaphora* is a sub-class of noun-phrase anaphora when the heads of the antecedent NP and the referring NP are lexically identical. *Indirect anaphora* occurs when they are not. In *Example 4.4 John Armstrong* and *Armstrong* (subscripted as $_{i1}$) denote one person and *Mlle. Giraud* and *the woman* (subscripted as $_{i2}$) denote another. The first case ($_{i1}$) is an example of *direct anaphora*. The second case ($_{i2}$) is an example of *indirect anaphora*.

In order to resolve noun-phrase anaphora, it is necessary to know that a definite description at hand it is indeed anaphoric (that is, it does not introduce a new entity). In order to do so, this work implements a set of rules proposed by Poesio and Vieira (2000). These rules distinguish definite NPs introducing new entities, direct anaphora and indirect anaphora. The system also implements a technique for resolving direct anaphora proposed in the same paper. The classification of definite NPs and the resolution of direct anaphora are described in **Section 4.3.5**.

Once the system has identified new discourse entities and has resolved direct anaphoric expressions, it attempts to resolve indirect anaphora. In order to do so, it uses the algorithm of Lappin and Leass (1994) already described in the previous section. **Section 4.3.5** explains why it was possible to use this algorithm to resolve indirect anaphora in this case.

Figure 4.5 shows the overall structure of the noun phrase anaphora resolution sub-module.

4.3.5. Classifying definite noun phrases

The noun-phrase resolution proceeds in several stages (see the pseudo-code in **Figure 4.3.5.**) At first, the program verifies whether a given definite NP is an occurrence of direct anaphora (line 5). If it is not, then it checks whether it introduces a new entity (line 9). If both these checks fail, the program assumes that the NP is an occurrence of indirect anaphora co-referring with a

Figure 4.5. Noun-phrase anaphora resolution.

definiteNPs – a data structure that holds all definite noun phrases found in a sentence
windowSize – the size of a window within which potential antecedents are considered
allCandidates – all potential antecedents

Input: allCandidates, windowSize, sentences, quotes, recencyDecayFactor

```
1 for each sentence in sentences do
2       choose all definite noun phrases and store them in definiteNPs
3       for each nounPhrase found in this sentence do
4               // check whether this NP is an example of direct anaphora
5               antecedent = IsDirectAnaphora ( nounPhrase, allCandidates )
6               if antecedent is not null
7                       continue
8               // check whether this NP introduces a new entity
9               if IsFirstMention ( nounPhrase ) is True
10                      continue
11              //otherwise it is an anaphoric noun phrase
12              //we use the same machinery as for pronouns
13              antecedent = FindAntecedent ( nounPhrase,  allCandidates)// see Figure 4.4.b
14      link nounPhrase and antecedent
15 def  IsDirectAnaphora (curNP, candidates)
16      curHead = nominal head of curNP
17      for each candNP in candidates do
18              isMatch = False
19              candHead = nominal head of candNP
20              if curHead == candHead
21                      if curNP is premodified
22                              if candNP is not premodified
23                                      isMatch = True
24                                      break
25                              //otherwise check agreement between premodifiers
26                              if premodifiers of curNP are a subset of premodifiers of  candNP
27                                      isMatch =True; break;
28              if isMatch == True
29                      antecedent = candNP
30                      curScore = calculate syntactic score of curNP
31                      candidates [ antecedent ] . score = curScore
32                      return antecedent
33      return null
```

previously encountered entity. The call to *FindAntecedent* on line 13 uses the same procedure as was used for pronominal anaphora to find the antecedent of the anaphoric NP.

Identifying direct anaphora. The procedure *IsDirectAnaphora* in **Figure 4.5** identifies occurrences of direct anaphora.

In order for a definite NP to be a direct anaphoric reference to a previously encountered entity, their nominal heads must be lexically identical (see *Example 4.4.*) However, this is not al-

ways a sufficient condition. For instance, *the young man* cannot be an antecedent of *the old man*. In other words, pre-modifiers of definite NPs can prevent co-reference. To deal with such cases, the program performs an additional check when looking for antecedents of pre-modified definite descriptions (line 21). A definite NP *N* can be a direct anaphoric expression referring to a candidate antecedent *C* iff 1) the pre-modifiers of *N* are a subset of the pre-modifiers of *C* or 2) *C* is not pre-modified.

If two NPs are co-referential (line 28), the score of the antecedent is adjusted to reflect the fact that it has been recently mentioned (line 30).

Identifying definite NPs that introduce new discourse entities. I have already mentioned that a significant percentage of definite NPs introduce new discourse entities. Poesio and Vieira (2000) report that 52% of all definite descriptions in their corpus introduce new entities. In order to identify such expressions, the system implements several heuristics proposed by Poesio and Vieira (2000). **Table 4.2** lists these heuristics.

Resolving indirect anaphora. When the program encounters a definite NP that is neither directly anaphoric nor introduces a new discourse entity, it attempts to resolve the NP using the same machinery as for personal pronouns (line 13 of **Figure 4.5**). The same procedure, *FindAntecedent*, is used for finding antecedents of both personal pronouns and indirect noun phrase anaphora. Since this is not a technique practiced in the community, it merits some explanation.

In general, the problem of resolving indirect anaphora is considered to be very challenging (Gelbukh and Sidorov 1999; Fan et al. 2005). As a consequence, most systems rely on semantic knowledge of one sort or another in order to find antecedents of such expressions: a semantic knowledge base (Fan et al. 2005), WordNet (Poesio and Vieira 2000) or a thesaurus (Gelbukh and Sidorov 1999). However, the system described in this dissertation deals with a small subset of indirect anaphoric expressions: it only attempts to resolve definite NPs that denote animate entities. This limitation simplifies the task considerably. In this case, I assume that a definite NP denoting a person can be co-referential with another NP, if both agree in gender and number. Because of this limitation and because the gender and number information is available, the resolution of indirect anaphora becomes rather similar to the resolution of pronominal anaphora. This is why it is possible to use the same machinery and not use any additional semantic resources. The only difference between the resolution of indirect anaphora and pronominal anaphora is in

Table 4.2. Heuristics for identifying new discourse entities.

New proper name.

If a definite NP is a mention of a proper name and is not a case of direct anaphora, then it introduces a new entity.

Unexplanatory modifiers.

If an NP is modified by a superlative adjective (e.g., the best, the tallest) or an ordinal number (the first, the second), it introduces a new entity.

The tallest woman spoke.

Restrictive modifiers.

a) Restrictive pre-modification. If a definite NP is pre-modified by a noun, an adjective, or a cardinal number, it introduces a new entity.
   The 9/11 victims
b) Restrictive post-modification: if an NP is post-modified by 1) a relative clause or 2) a prepositional clause that is not separated by commas from the rest of the sentence, this NP introduces a new discourse entity.
   The woman with whom we spoke is a writer.

Appositions.

If an NP is modified by an apposition, it introduces a new entity.

Old Anthony Rockwall, retired manufacturer and proprietor of Rockwall's Eureka Soap, looked out the library window of his Fifth Avenue mansion and grinned.

Copular constructions.

NPs in copular constructions introduce new entities.

The manager is John. (or *John is the manager*).

handling of cataphoric expressions: the system does not handle the occurrences of indirect noun phrase cataphora because of the possibility of introducing cycles in the graph of referring expressions and antecedents.

4.3.6. Evaluation of the anaphora resolution module

As a rule, in order to evaluate the performance of an anaphora resolution algorithm, a researcher compares it against a gold standard created by humans. Several people annotate test texts for the presence of anaphoric expressions and manually find an antecedent for each expression. Performing such an evaluation thoroughly is labour-intensive and not trivial: (Poesio and

Vieira 2000) report than two annotators working on 14 texts exhibited very little agreement when identifying cases of indirect anaphora: the agreement kappa (Cohen 1960) was only 0.24.

Table 4.3. Results of anaphora resolution.

Type of anaphora	All	Correct	Incorrect	Error rate, %
Pronominal	597	507	90	15.07
Nominal	152	96	56	36.84
Both	749	603	146	19.49

A thorough evaluation of the anaphora resolution module would be prohibitively labour-intensive. In addition, the module plays a purely practical role and is not a research direction in itself. This is why I estimated its performance by manually verifying the results achieved on two short stories of the training set (**Table 4.3)**[27]. These results compare quite favorably with the ones reported in literature (see **Section 4.6**). The error rates of pronominal anaphora resolution are significantly lower than those of noun phrase anaphora resolution (15.07% vs. 36.84%). This is not unexpected because resolving noun phrase anaphora is known to be a very challenging task. The results also reveal that referring to characters by pronouns is much more frequent than by nouns – in this case, the ratio of pronominal to nominal expressions is almost 4:1. This suggests that resolving pronominal anaphoric expressions is crucial to summarizing short stories.

4.4. Identification of important characters

Not all mentions of animate entities refer to important characters. Some of them refer to episodic characters, some are metaphoric, some are generic, *etc*. In order to identify characters that are central to the stories, I used normalized frequency counts.

Formula 4.1 $N_{char} = F_{char} / T$

In *Formula 4.1*, N_{char} is the normalized frequency of character *char* in the story, F_{char} is the number of times this character is mentioned and T is the total number of tokens in the story. The value of F_{char} reflects both direct and anaphoric mentions of *char*.

After a small study of the stories in the training set, I observed that there exists a value of N_{char}, which separates important characters from unimportant ones. In other words, characters with normalized frequency equal to or above this value tend to be important, while all others –

[27] The stories are different from the ones used for fine-tuning.

unimportant. I selected this value *A* as a cut-off point for identifying important characters. The value of *A* was fixed at 0.005.

4.5. Conclusion

This chapter contains a description of the part of the system responsible for recognizing important entities in short stories. At the end of this stage of processing, the stories are annotated for the presence of locations and important characters. In addition, the majority of anaphoric mentions of characters are resolved and this information is made available to other modules.

The preliminary experiments on the training part of the corpus revealed a few interesting traits of short stories. It appears that explicit temporal information is scarce in short fiction and that authors use other means to tell a reader when a story takes place. In addition, pronominal references to characters are much more common in fiction than nominal ones. This suggests that the resolution of pronominal anaphora is crucial when summarizing this genre of data.

Once the system has identified important entities in the stories, it proceeds to select summary-worthy sentences and compose summaries out of them. This stage is described in **Chapters 5** and **6**.

The last section of this chapter provides a brief overview of the state-of-the-art in anaphora resolution technology and explains why I chose the algorithm of Lappin and Leass (1994) and not any other.

4.6. Anaphora resolution: related work

In this brief review I make a distinction between pronominal anaphora resolution and nominal anaphora resolution. These two groups vary significantly in terms of type and depth of analysis as well as in terms of success rate.

Pronominal anaphora resolution. Many of today's approaches to pronoun resolution are indebted to the algorithm of Lappin and Leass (1994). This is especially true of the motivation behind the salience weighting factors from **Table 4.1.** As the reader will see, these are echoed in a number of subsequent works (Kennedy and Boguraev 1996; Mitkov 2002).

Kennedy and Boguraev (1996) propose a system that extends the algorithm of Lappin and Leass (1994). Its main advantage is the fact that it does not require in-depth syntactic parsing. Instead, the system uses the output of an ENGCG part-of-speech tagger (Karlsson et al. 1995)[28]. The authors implement a simple grammar to identify NPs and order them by the offset from the start of the text. The pronoun resolution procedure is similar to that of Lappin and Leass (1994). By lifting the requirement of complete syntactic parsing, the authors make pronoun resolution more available to the community[29]. They evaluate their system on 27 texts of different genres (news, product announcements, web pages) and report a success rate of 75%.

Another syntactically-motivated approach to pronoun resolution is MARS system (Mitkov 2002). In order to obtain syntactic information, MARS uses the Connexor parser (Tapanainen and Jarvinen 1997.) The algorithm assigns a score to each potential antecedent based on a list of preferences called *antecedent indicators*. The antecedent indicators reward certain types of NPs, such as the first noun phrase in a sentence, nouns that occur frequently in the same paragraph as the pronoun, those that occur in section headings or in certain syntactic constructions and nouns that co-occur with a pre-defined set of verbs. MARS also rewards pronominal antecedents of other pronouns[30]. It ensures that an antecedent and a referring expression agree in number and gender and that they satisfy a set of syntactic constraints similar to that of Lappin and Leass (1994). An antecedent with the highest score that satisfies both conditions is linked to the pronoun. The author evaluates MARS on a corpus of computer manuals and reports the accuracy of 63.68%.

Ge et. al. (1998) propose a different approach. The authors use training and testing corpora and estimate several probabilistic factors that influence the presence of a co-referential link between an NP and a 3rd person pronoun. They consider the following factors: the distance[31] between the pair, gender, number and animacy of the proposed antecedent, syntactic parallelism (*i.e.*, both the candidate antecedent and the pronoun have the same syntactic function) and the frequency of the candidate antecedent in the corpus. These probabilities are multiplied to yield a

[28] It should be noted that this tagger belongs to the category of so-called *super-taggers*. The tagger provides not only a part-of-speech tag for each token, but also its syntactic function.
[29] The title of this paper is *Anaphora for Everyone*.
[30] See pages 147-149 and 165-168 of (Mitkov 2002) for a complete list of the antecedent indicators.
[31] The meaning of the term *distance* is not straightforward here. In order to compute the *distance* value, the authors run the algorithm proposed by Hobbs (1978) which selects up to *N* possible antecedents for a pronoun in the order of preference. The value of *distance* equals the index of a candidate antecedent in the list of *N* possible antecedents.

single probability value and the system selects the antecedent with the highest value. The algorithm achieves 84.2% accuracy on a subset of Penn Treebank (Marcus et al. 1993). (The corpus contains 93,931 words and the authors manually tag it for the presence of pronouns and their correct antecedents).

I selected the works of Boguraev and Kennedy (1996), Mitkov (2002) and Ge et. al. (1998) for a review because these systems represent important trends that emerge when one studies the approaches to pronoun resolution. In order to find the most likely antecedent for a pronoun, researchers rely on syntactic information, shallow lexical knowledge (co-occurrence patterns and frequency counts) and shallow semantic knowledge (such as dictionaries containing gender information about proper nouns). The immediate antecedent selection can be performed in several ways: by using manually designed rules (Lappin and Leass 1994; Mitkov 2002), probabilistic modeling (Ge et al. 1996) or machine learning (Modjeska et al. 2003.)

Resolving noun phrase anaphora. The resolution of indirect noun phrase anaphora is considered to be a very difficult problem that requires semantic knowledge to be solved satisfactorily.

Poesio and Vieira (2000) use the WordNet ontology (Fellbaum 1998) to approximate semantic knowledge. Their system identifies the occurrences of indirect anaphora by filtering out direct anaphora and NPs that introduce new entities. For each indirectly anaphoric NP, the system verifies whether there exists a WordNet relation between it and a potential antecedent NP. The search for potential antecedents is restricted to a window of five sentences preceding the anaphoric NP. If such a relation is found, the pair is judged to be co-referential. The system achieves 62% F-score in identifying indirect anaphora on the test set and 28% accuracy in assigning correct antecedents on the training set[32].

Fan et al. (2005) use WordNet as a knowledge base to find antecedents of indirect anaphoric expressions. The authors use an interpreter that performs two searches for each potential referring expression – antecedent pair. Given a referring expression *R* and a candidate antecedent *A* the interpreter 1) starts at *R* and searches for *A* or a sub-class or a super-class of it and 2) starts at *A* and searches for *R* or a sub-class or a super-class of it. The shorter of the 2 paths is returned. If

[32] The main direction of the research described in (Poesio and Vieira 2001) distinguishing between definite NPs introducing new entities and direct and indirect anaphora. The authors do not report the accuracy of finding correct antecedents on the testing set.

there is more than one candidate antecedent, the system chooses the shortest of the candidate paths, modeling preference for closer semantic relations.

Bunescu (2003) uses the Web to approximate semantic knowledge. Instead of using a lexical knowledge base, this system approximates the likelihood of co-reference between a pair of NPs by mining the Web for template-like patterns of their co-occurrence. The NPs that co-occur frequently are more likely to be co-referential. Relying on the Web instead of a lexical resource makes it possible to apply this algorithm to languages for which no such resources are available.

Soon et al. (2003) use machine learning to find the most likely antecedent of a referring expression (the referring expression may be both nominal and pronominal). The system computes a feature vector for each potentially co-referential pair. These features include the distance between the candidates, whether the candidates are pronominal, whether the heads of two candidate NPs are identical, gender and number agreement, semantic class agreement[33] and a few others. The authors train and test a C5.0 classifier (Quinlan 1992) on the data used in Message Understanding Conference-6 and Message Understanding Conference -7. The system achieves recall of 58%, precision of 67.3% and F-score of 62.6%.

Anaphora resolution in this dissertation. The anaphora resolution module in this system plays a practical role. Its purpose is to increase the amount of information available about characters. Therefore, it was reasonable to implement the algorithms that do not require annotated data for training or for probabilistic modeling[34]. This is why I chose to implement the algorithms of Lappin and Leass (1994) and Poesio and Vieira (2000). I had access to a broad-coverage syntactic parser (*i.e.,* the Connexor parser), which allowed me to use the approaches that require deep syntactic knowledge. The algorithm of Lappin and Leass (1994) influenced many of the subsequent approaches and it seemed to be a natural choice. As for the rules adopted from Poesio and Vieira (2000), I am not aware of any other work proposing such thoroughly motivated and clearly outlined rules for identification of various types of definite noun phrases.

---

[33] (Soon et al. 2003) define a small set of pre-defined semantic classes: *male*, *female, location, organization, date, time, money, etc.* They manually map each class to a correct WordNet entry (including he correct sense). The system assigns a correct semantic class by verifying that a WordNet entry for a candidate noun is a sub-class of an entry for one of the semantic classes.

[34] This is especially true because I am not aware of any corpus of fiction annotated for the presence and antecedents of anaphoric expressions.

Chapter 5. Using Aspect to Identify Descriptive Sentences

5.1. Chapter overview

The summarization system described in this dissertation produces summaries of short stories in two stages: identifying important entities in each story and selecting descriptive sentences that focus on such entities. **Chapter 4** describes the process of identifying important entities. **Chapter 6** explains in detail the sentence selection process. This chapter provides background information on linguistic concepts central to the sentence selection process.

In order to separate descriptive sentences from those which relate events, the system collects information about a number of properties for each clause of every sentence in a story. The properties of interest are those that help approximate a particular semantic property of a clause - its aspectual type.

This chapter presents the notion of aspect and discusses what properties of a clause signal it. **Section 5.2** defines aspect and presents the most common types of clauses based on the standard aspectual hierarchy. **Section 5.3** lists and explains what properties of a clause may signal its aspectual type. **Section 5.4** provides a brief overview of the body of research in computational linguistics that focuses on determining aspect automatically.

5.2. Aspect: the concept and fundamentals

The term *aspect* refers to "different ways of viewing the internal temporal consistency of a situation" (Comrie 1976, p. 3). It can also be explained as the property of a clause that gives one an idea about the temporal flow of an event or state being described and about the position of a speaker with respect to it. Despite the fact that aspect has been extensively studied, the literature in linguistics, computational linguistics and philosophy exhibits little agreement as far as terminology is concerned: the same phenomenon is also known as *situation type* (Huddleston and Pul-

lum 2002; By 2002), *verb aspect* (Dowty 1979), *Aristotelean aspect* (Binnick 1991) and quite a few others. In this dissertation I employ two terms in an interchangeable manner: *aspect (*or *aspectual type)* and *situation type*.

Figure 5.1. Aspectual classification hierarchy (Huddleston and Pullum 2002, p. 118.)

I. **States** (stative situations)
  1. *He believes in God.*
  2. *They understand each other very well.*
II. **Events** (dynamic situations)
  1. **Processes** (durative events)
    a) **Activities** (atelic durative events)
      3. *He is playing golf.*
      4. *They talked on the phone.*
    b) **Accomplishments** (telic durative events)
      5. *He is writing an essay.*
      6. *He read that book in an hour.*
  2. **Achievements** (punctual)
      7. *He dropped the key.*
      8. *She stumbled.*

Aristotle in his work *Metaphysics* laid ground for distinguishing different types of sentences based on their situation (Aristotle). He categorized verbs depending on whether their meaning inherently implied a conclusion. Aristotle distinguished between *movements* - verbs whose meaning had a completion component (e.g. *learn*, *build*, *move*) - and *actualities* - verbs whose meaning did not have such an implication (e.g. *understand*, *see*, *think*). This classification does not directly correspond to any other in use today, but it was the first attempt to classify situations based on their internal temporal consistency.

While the definition of the term *aspect* is hardly self-explanatory, it is easily demystified by examples. **Figure 5.1** shows the aspectual hierarchy based on Huddleston and Pullum (2002, p. 118).

The first distinction is between *states* and *events*. *Events* are processes that go on in time and consist of successive phases (Vendler 1967, p. 99). For instance, an event of writing an essay consists of writing separate words, correcting, pausing between words, *etc*. A *state* of understanding each other, on the other hand, does not imply such compositionality: it remains unchanged through out the whole period when it is true. In other words, the meaning of events exhibits a dynamic component, while that of states - does not.

Events are further categorized based on whether a particular situation lasts for some time or occurs momentarily. Atomic events are referred to as *achievements*, while events that imply duration are known as *processes*. For instance, the nature of events such as dropping, stumbling,

recognizing, *etc.* is such that they occur instantaneously and, as such, present examples of achievements. On the other hand, events such as playing golf or writing an essay are not atomic in nature and last for some time. Such events are examples of processes.

Processes are classified into *accomplishments* and *activities* depending on whether a situation implies an ending (Vendler 1967, p. 100). This property is known as *telicity*. Reading a book (in the context of example 6 in **Figure 5.1**) implies that the person finished reading it and the overall situation is telic. We cannot say that he has read the book in the first 15 minutes of doing so because the implied ending was not achieved (the book was not read). Such situations are referred to as *accomplishments*. On the other hand, playing golf or talking on the phone is true throughout the whole period of a person doing so. The situation implies no conclusion and is atelic. Such situations are called *activities*.

It is extremely important to clarify at an early stage that the term *aspect* as used in this dissertation refers to a semantic property of a clause and not to a particular syntactic form, *i.e.*, not to the grammatical aspect. Throughout this dissertation the term *aspect* refers to a property of the meaning of the clause. Grammatical aspect, on the other hand, is a classification of different conjugated forms of a verb and is independent of context. When discussing the distinction between *I am talking* and *We have talked* in terms of grammatical aspect, one can say that the former is progressive and the latter is perfective. When looking at aspect of these examples, one observes that both are of type activity.

There exists a further distinction in the aspectual classification: between *singular* and *multiple* situations. All sentences used as examples in **Figure 5.1** are instances of singular situations, that is, an event or a state which is described in a sentence occurs only once. Multiple situations, on the other hand, are situations in which the central event or state occurs more than once. The types of multiple situations based on the classification by Huddleston and Pullum (2002, p. 123) are displayed in **Figure 5.2**. Multiplicity of a situation affects its overall type. For instance, looking at the example *She always drops things* in

Figure 5.2. Types of multiple situations (Huddleston and Pullum 2002, p. 123.)

I. **Iterative**.
He knocked on the door.

II. **Repeated**.
We talked three times.

III. **Serial**.
She always drops things.

Figure **5.2**, one sees that although every singular instance of her dropping a thing is an achievement, the overall situation type of this sentence is a state.

By looking at examples in **Figures 5.1** and **5.2** one may notice that the situation type of a clause is related to whether a clause is descriptive or whether it relates events - a very important distinction in the context of this work of the need to identify descriptive sentences focusing on important entities. Intuitively, singular states are more likely to describe states of affairs than other types of situations. On the other hand, serial situations tend to refer to things that were happening for an extended period of time. By learning to identify such clauses (state clauses and serial situations) I expect to identify descriptive sentences and use them for creating summaries.

The rest of this chapter discusses properties of clauses that signal the aspect of a clause with a particular emphasis on the properties that can be established automatically. **Chapter 6** describes how I use those properties to identify sentences that set out the background of a story.

5.3. Properties of a clause that signal its aspectual type

5.3.1. Overview

The contemporary literature in linguistics and computational linguistics available today points out a number of surface markers and semantic tests that are correlated with the aspectual type of a clause. Some of the more notable ones are the lexical aspect of a verb, temporal expressions, tense and a few others. This section surveys the properties of a clause that signal its aspectual type with a particular emphasis on those that can be established automatically using the state-of-the-art NLP technology.

5.3.2. Lexical aspect vs. aspect of a clause

The term *aspect* refers to a particular property of meaning of a clause - it provides one with information as to whether the overall situation described in a given clause is dynamic, durative or telic. A corresponding property of the main verb in a clause is referred to as the *lexical aspect* of the verb. Lexical aspect refers to a property of the verb viewed in isolation, without regard to a

context provided by a particular clause. According to the classification described by Dorr and Olsen (1997) a verb may be a *state* (or *stative*) *verb* (*e.g.*, *believe*), or an *event verb* (*e.g.*, *run*). Just as it is the case with clauses, event verbs are further subdivided into *activities* (*e.g.*, *read*), *accomplishments* (*e.g.*, *take tes*t) and *achievements* (*e.g., drop*).

Lexical aspect is not an absolute property of a verb, at least not in all cases. While some verbs exhibit a strong preference to form clauses of a particular type, others are highly ambiguous with respect to aspect. Consider a state verb *believe* in examples *5.1a* and *5.1b*. The verb exhibits certain reluctance to form event clauses, as can be seen from example *5.1b* (**He was believing her*[35] is not a grammatical sentence). Examples *5.2a* and *5.2b*, on the other hand, exemplify the opposite situation: an achievement verb *kill* successfully participates in both achievement and activity clauses (examples *2a* and *2b* respectively).

Example 5.1a. *He believes in ghosts.*

Example 5.1b. **He was believing her.*

Example 5.2a. *He killed a mosquito.*

Example 5.2b. *This headache is killing me.*

Example 5.2c. *He dropped at least three glasses a day during every summer when he was small.*

An extreme case of ambiguity with respect to aspect is shown in example *5.2c*: the overall situation is a state, but it can be decomposed recursively into situations of other types, all the way down to *dropping a glass*, where the main verb *drop* has the lexical aspect of type *achievement*. This phenomenon (*i.e.*, situations where the lexical aspect of the main verb is altered by other constituents of the clause) is called *type-shifting* (By 2002, p. 36).

In the real world usage the examples such as *5.2b* are much more frequent than those similar to *5.2c*. This is why knowing the lexical aspect of the main verb is very helpful in establishing the situation type of the clause. In addition, the relation between the lexical aspect of a verb and the aspect of a clause has been discussed by a number of researchers. In particular, Dorr and Olsen (1997) have proposed a privative model of this relation (see **Table 5.1**). According to the

[35] An asterisk (*) denotes incorrect or marginally correct usage.

Table 5.1. Privative featural identification of aspectual classes (Dorr and Olsen 1997.)

Aspectual class	Telic	Dynamic	Durative	Examples
State			+	know, believe
Activity		+	+	paint, walk
Accomplishment	+	+	+	destroy
Achievement	+	+		notice, win

model, the verbs are categorized into aspectual classes based on whether they exhibit one or more of the following properties: dynamicity, durativity and telicity. Dorr and Olsen speculate that, depending on the context of usage, verbs may form clauses that have more of these properties than the main verb viewed in isolation, but that it is impossible for a verb to 'shed' one of its properties. Examples *5.3a* and *5.3b* exemplify the matter. In the example *5.3a* a state verb *know* participates in an accomplishment clause; the clause is telic, while the verb by itself is not. On the other hand, an attempt to deprive the accomplishment verb *destroy* of its telic meaning and to construct a clause of type activity fails to produce a grammatical clause (example *5.3b*).

Example 5.3a. *He knew it that very moment. (accomplishment)*

Example 5.3b. **He was destroying it for an hour. (activity)*

That is why knowing the lexical aspect of a verb is very important when attempting to establish the situation type of a clause. The summarization system (described in **Chapter 6**) considers the lexical aspect of main verbs when selecting summary-worthy sentences.

5.3.3. Tense and grammatical aspect

The grammatical forms of tense and aspect used in a particular clause place a number of constraints on its situation type. Dowty (1979, p. 55) and Huddleston and Pullum (2002, p. 119) stipulate that progressive tenses are not normally used in state clauses (see examples *5.4a* and *5.4b* for illustrations). According to Huddleston and Pullum (2002, p.121), achievement clauses also cannot be realized using progressive tenses (see example *5.4c*).

Example 5.4a. *John is running.*

Example 5.4b. **John is knowing the answer.*

Example 5.4c. **John was recognizing her.*

It should be noted that this rule is often violated in the contemporary usage of English, especially in the informal setting (see example *5a*). However, this linguistic development is of little concern to us because the corpus at hand consists mostly of XIX century fiction.

Example 5.5a. *I'm loving this weather.*

Among the constraints imposed by the grammatical tense is the special relation between simple present tense and event clauses. As a rule, clauses that are realized in simple present tense cannot denote events, but only states (Huddleston and Pullum 2002, p. 119). The matter is illustrated through examples *5.6-5.8*. We can see that simple present tense combines freely with state clauses (examples *5.6b* and *5.8b*). In the case of example *5.8b*, the usage of present tense promotes interpreting the overall situation type of the clause as a state (an event interpretation is possible but unlikely) despite the fact that the main verb *dance* is an event verb. In the case of example *5.7b* one may note that using present tense in an event clause produces an awkward clause, which borders on ungrammatical.

Example 5.6a. *She knew history well.*

Example 5.6b. *She knows history well.*

Example 5.7a. *She fell off a chair.*

Example 5.7b. **She falls off a chair.*

Example 5.8a. *She danced (last night).*

Example 5.8b. *She dances a lot.*

Perfect tenses also place a number of constraints on the situation type of a clause. Siegel (1998, p. 50) points out that clauses realized in present, past or future perfect tenses may never relate activities as perfect tenses tend to imply a completion of an event or a state that is described (see example *5.9a*). This restriction does not apply to perfect progressive tenses (example *5.9b*).

Example 5.9a. **I have painted.*

Example 5.9b. *I have been painting.*

Due to the limitations listed above, knowing tense of the clause may provide one with invaluable information about its situation type. This is especially true if a clause is realized in simple present tense or in a progressive tense.

Table 5.2. Types of temporal expressions (Harkness 1987.)

Type	Examples
Location	In 1999, today, two days ago
Duration	since then, within two hours, for twenty minutes
Frequency	often, occasionally, twice, every day
Enactement	never

5.3.4. Temporal expressions and aspect

Temporal expressions, such as *now*, *yesterday*, *a minute ago, etc.* are wide-spread in English. In many cases such expressions unambiguously signal the situation type of a clause (see examples *5.10a-5.10c*).

Example 5.10a. She always gets up late. (state)

Example 5.10b. She instantly woke up. (achievement)

Example 5.10c. She read the chapter within 10 minutes. (accomplishment)

The linguistic literature (Harkness 1987) suggests four basic types of temporal expressions: *location* expressions, expressions of *duration*, expressions of *frequency* and *enactment* expressions. **Table 5.2** shows examples of expressions of each type.

The relation between temporal expressions and the situation type of a clause has been extensively studied (Siegel 1998, p. 53-55), (By 2002, p. 19-25), (Harkness 1987). There seems to be a unanimous acceptance of the fact that temporal expressions influence the situation type of clauses, yet there exists no clear or elegant set of rules. Some of the generally accepted rules are listed in the following paragraphs, but it should be clear that the relation between temporal expressions present in a clause and its aspectual type extends far beyond these examples.

By (2002, p. 21) points out that there exists "a natural connection between units of time used in a temporal expression and the typical length of the event under discussion". For instance, if one wants to find out how many hours a race lasted, a listener may presume that it lasted less than a day, or less than a few days (see examples *5.11a* and *5.11b*).

Example 5.11a. The compilation lasted 100 seconds.

*Example 5.11b. *The war lasted 1,000,000 seconds.*

Siegel (1998, p. 53) states that frequency expressions usually modify only telic events (achievements and accomplishments).

Example 5.12a. John glanced at his watch repeatedly. (achievement)

*Example 5.12b. *John stared at the president repeatedly. (activity)*

Quirk et al. (1985, p. 177-178) stipulate that the state clauses are not easily modified by frequency expressions because it makes no sense to pluralize them.

Example 5.13a. He often turned to see whether anyone was following him. (activity)

*Example 5.13b. *He often knew history. (state)*

Huddleston and Pullum (2002, p. 122) state that expressions of duration are not feasible with achievements because of the punctual nature of the latter (see examples *5.13a* and *5.13b*).

Example 5.14a. He collapsed that very moment. (achievement)

*Example 5.14b. *He collapsed for an hour. (achievement)*

Dowty (1979, pp. 55-60) stipulates that accomplishment and achievement clauses combine easily with temporal expressions of duration consisting of prepositional phrases with *in* (e.g., *in an hour*, *in a year*), but almost never allow expressions consisting of prepositional phrases with *for* (e.g., *for an hour*, *for a year*). On the other hand, activity clauses combine naturally with for-expressions and awkwardly with in-expressions. Examples *5.15, 5.16* and *5.17* illustrate these restrictions for achievement, accomplishment and activity clauses respectively.

Example 5.15a. He won the race in a hour. (achievement)

*Example 5.15b. *He won the race for an hour. (achievement)*

Example 5.16a. He read the chapter in an hour. (accomplishment)

*Example 5.16b. *He read the chapter for an hour. (accomplishment)*

Example 5.17a. He walked for an hour. (activity)

*Example 5.17b. *He walked in an hour. (activity)*

It is clear that the knowledge about temporal expressions present in a clause is valuable for determining its aspectual type. Yet, it is also evident that capturing such expressions automati-

cally is by no means trivial. **Chapter 6** describes how the summarization system captures and uses such information.

5.3.5. Other indicators of aspect

Lexical aspect, tense, grammatical aspect and temporal expressions are some of the most important indicators of the situation type of a clause. Several other properties of a clause also signal its aspect, even though there is less agreement on the manner of this influence.

Siegel (1998, p. 51) points out that sentences without a deep subject (*i.e.,* imperative sentences and those realized in passive voice) are constrained to appear with events (see examples *5.18a-5.18b*). While this may be true in most cases, example *5.18c* shows a counter-example.

Example 5.18a. His watch was lost. (event)

Example 5.18b. Read! (event)

Example 5.18c. He was well known. (state)

In cases where the main verb takes a direct object (*e.g.*, *he read a book*), properties of the direct object may also influence the interpretation of the clause, as can be seen from examples *5.19a* and *5.19b*. Example *5.19a* promotes a state interpretation, *i.e.* that he usually read books. Example *5.19b,* on the other hand, is more likely to be interpreted as an event.

Example 5.19a. He read books.

Example 5.19b. He read that book.

The linguistic indicators of the situation type of a clause that I have explained so far are surface indicators: that is, they have syntactic or morphological presence in a clause and can be "present or absent". The linguistic literature offers many more tests that help establish aspectual type of a clause that are more semantic in nature. This work does not seek to list those exhaustively, as such tests cannot be applied automatically. However, some of the most important ones are listed below.

Dowty (1979) stipulates that if a given clause can be paraphrased using a pseudo-cleft construction with *do*, then a clause denotes an event, not a state. This test is illustrated through examples *5.20a* and *5.20b*. This test is also accepted by Huddleston and Pullum (2002).

Example 5.20a. He ran away. (event) - What he did was run away.

*Example 5.20b. He knew the rules. (state) - *What he did was know the rules.*

Another test proposed in the same source consist of rephrasing a clause in question so that the main verb becomes a complement of a verb *force* or *persuade*. Only event clauses can be successfully rephrased in this manner and state clauses resist this test (see examples *5.21a* and *5.21b*). Along the same lines, only event clauses can co-occur with the adverbs *deliberately* or *carefully* (examples *5.22a* and *5.22b*).

Example 5.21a. He did the homework (event) - He was forced to do the homework.

*Example 5.21b. He knew history (state) - *He was forced to know history.*

Example 5.22a. He did the homework carefully.

*Example 5.22b. *He knew history carefully.*

The set of rules and tests for establishing aspect listed in this section (*i.e.,* **Section 5.3**) is by no means complete, but it contains the most authoritative tests that can be applied in an automatic manner. More tests can be found in the works of Vendler (1967), Dowty (1979), Siegel (1998), By (2002) and Huddleston and Pullum (2002).

5.4. Establishing aspect automatically

The aspect of a clause is a semantic property and, as such, it is subject to numerous exceptions and depends on the context of usage. Almost every rule listed in this section has exceptions, some of which have been explicitly exemplified (*e.g.,* using stative verbs in progressive tenses). In addition, the interpretation of a clause may be altered when more that one indicator of aspect appear together or by the context in which a clause appears (see examples *5.23a*-*5.23c*).

Example 5.23a. She immediately blushed when someone addressed her when she was young. (state)

Example 5.23b. *It was time to take a break: she read a lot (tonight). (activity)*

Example 5.23c. *She read a lot when she was young. (state)*

In other words, the nature of the aspect of a clause is compositional and often does not obey strict rules. Establishing the aspect of a clause is relatively easy for humans; yet it is extremely difficult to perform this classification automatically. A number of researchers have attempted to tackle this problem, such as Siegel (1998), Dorr and Olsen (1997) and, to some degree Merlo et al. (2002) and Korhonen (2000).

Siegel (1998) proposes a fully automatic approach to determining lexical aspect for a set of verbs as used in a particular domain. The work concentrates on classifying verbs as opposed to clauses because the author takes a stance that verbs tend to form clauses of a single type (the same as lexical aspect of a verb) when used within a particular domain. The author classifies verbs according to their lexical aspect using supervised and unsupervised learning methods. The classification process relies on a number of surface indicators for each occurrence of a verb in the corpus, such as the presence of certain temporal adverbs, tense in which a clause is realized, presence and absence of a deep subject, *etc*. The results reported in this work are encouraging (93% accuracy when distinguishing stative verbs from event verbs in a corpus of medical discharge summaries), yet the methodology is not directly applicable in our case. The corpus used in this work (a collection of works of fiction) is not a domain-dependent corpus and it is unreasonable to expect that verbal aspect will directly translate into the aspect of a clause.

Dorr and Olsen (1997) develop an extended lexicon of verbs (a database of Lexical Conceptual Structures or LCS), which contains a number of semantic and syntactic properties for each verb, such as sub-categorization frames, semantic class of a verb according to the classification proposed in (Levin 1993) and a number of others. This valuable resource also contains information as to whether a verb is dynamic, durative and telic - the three properties that determine its lexical aspect. The authors extend their framework to allow determining the aspect of a clause, but this extension requires a very deep semantic analysis of a clause, which is not available in our case.

The more recent works of Korhonen (2000) and Merlo et al. (2002) do not tackle the problem of aspectual classification directly but attempt to solve a related problem: automatically categorizing verbs into a number of semantically-oriented classes (in both cases, the proposed classes

are based on the classification of Levin (1993)). Although both approaches are rather successful, they are not directly applicable in our case for the same reason as that of Siegel (1998): the proposed solutions establish properties of verbs viewed in isolation, while I am interested in determining the corresponding properties of clauses.

5.5. Conclusion

This chapter introduced the aspectual classification of clauses and discussed how this information may be leveraged for automatic summarization of short stories. It appears that singular state clauses and serial clauses are likely to set out the background of the story because their meaning exhibits a descriptive component. On the other hand, singular dynamic clauses are likely to relate the events.

Various properties of a clause signal its aspectual type. Some of the most important of these are the lexical aspect of the main verb, tense, the grammatical aspect of a clause and temporal expression found in the clause. These properties can be established automatically using the state-of-the-art NLP tools. By computing these properties, it is possible to approximate the aspectual type of the clause and to select descriptive clauses that focus on important entities.

The following chapter (**Chapter 6**) explains how these aspectual indicators are used to produce indicative summaries of short stories.

Chapter 6. The Description of the Sentence Selection Module

6.1. Chapter overview

This chapter describes the sentence selection component of the system for summarizing short stories. The system creates extractive summaries that help readers form adequate expectations about a story without revealing its plot. It functions by selecting sentences that satisfy two conditions: they focus on important entities and they relate the background of the story rather than events.

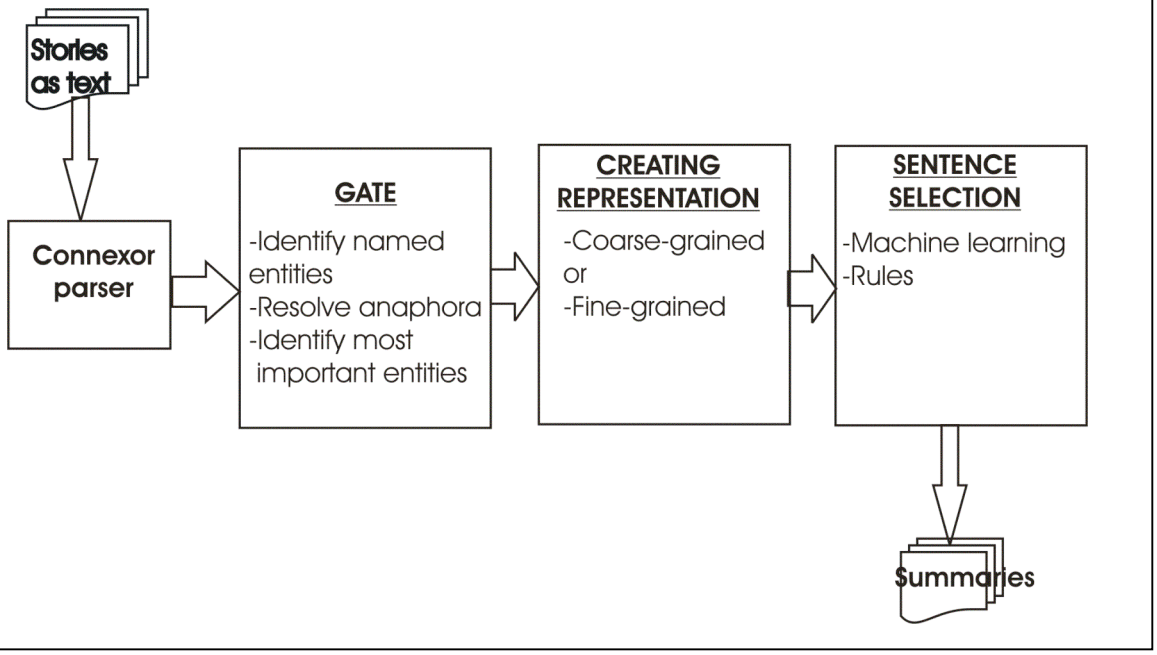
Section 6.2 provides an overview of the system and describes the options that it offers. **Section 6.3** discusses the representation of clauses, which forms the basis for the decision-making process. **Section 6.4** explains how the clauses with the main verb *have* are different from all others and how the system handles such clauses. **Sections 6.1 – 6.3** discuss two distinct sentence selection procedures that are available as a part of the system: machine learning and a set of manually designed rules.

6.2. Overall system design

Chapter 4 contains the description of the pre-processing component of the system, which annotates the original stories for the presence of important entities: locations and characters. It also establishes which characters are central to the story by taking into account pronominal and noun phrase anaphoric references. The next stage involves selecting salient background sentences and composing summaries out of them. This stage is describes in this chapter. **Figure 6.1** shows a high-level overview of the system.

Several system components are responsible for selecting salient background sentences. The stories, annotated for the presence of important entities (as outlined in **Chapter 4**), are parsed

Figure 6.1. System architecture.



with the Connexor Machinese Syntax parser. The sentences are then recursively split into *clauses* based on the results of parsing. For the purposes of this dissertation, a *clause* is defined as a main verb with all its complements, including subject, modifiers and their constituents.

Each clause is then represented as a vector of features describing its characteristics. The system offers two options with regard to clause-level representation: a fine-grained representation or a coarse-grained one. The main difference between these two representations is in the number of features and in the cardinality of the set of possible values, but not so much in what kind of information they carry. For instance, a fine-grained feature vector has three different features with seven possible values to carry tense-related information: *tense*, *is_progressive* and *is_perfect*, while a coarse-grained one carries only one binary feature*, is_simple_past_or_present*. Originally, only the fine-grained representation was designed and used. The coarse-grained representation was added later due to expectations that reducing the number of features and the cardinality of the set of values would facilitate and improve the sentence selection process.

In the next step, the system selects salient descriptive sentences. Regardless of the representation used, one may choose between two different procedures for sentence selection. The first

procedure employs machine learning techniques, namely the C5.0 decision tree induction (Quinlan 1992). The second procedure applies a set of manually created rules that guide the classification process. **Section 3** provides the description of features used in each dataset. **Sections 5.1 – 5.3** describe the experimental setting and **Chapter 7** presents the results.

The part of the system that selects descriptive sentences is implemented in Python.

6.3. Feature selection: description and motivation

Features for both representations are selected based on one of the following criteria:

(Criterion 1) *a clause should "talk" about important things, such as characters or locations*

(Criterion 2) *a clause should contain background descriptions rather then events*

Table 6.1 shows the number of features providing information towards each criterion, as well as the number of possible values. Both datasets have two continuous features and the values of these features are not included in *Number of Values* column in **Table 6.1**. The rest of the features are discrete. **Appendix C** contains a complete list of features used in both representations and the cardinality of the sets of possible values.

The features contributing towards *Criterion 1* can be divided into *character-related* and *location-related*.

The character-related features are designed so as to help identify sentences that focus on characters, not just mention them in passing. Usually, such sentences contain at least one mention of an important character with a salient grammatical function (*e.g.*, subject). Consequently, character-related features describe whether a clause contains a character mention and what its grammatical function is (subject, object, indirect object or other). Mentions of characters occurring early in the text tend to contain more salient background information. That is why character-related features reflect the position of a parent sentence relative to the sentence where the charac-

ter is introduced. In addition, these features capture the presence of a character mention that is modified by a noun phrase. The interest in noun phrase-modified character mentions is inspired by the fact that such constructions (*i.e.,* appositions) often introduce new entities into discourse (Poesio and Vieira 2000). For the same reasons, the system also establishes whether a character mention is nominal or pronominal (*e.g.*, *Jack* versus *he*), whether it is used in the genitive case (*e.g., Jack's*) and, for common nouns, whether the mention is accompanied by an indefinite article.

In discourse such as fiction, not all tokens that GATE gazetteer recognizes as markers of location denote locations. Location-related features help identify mentions of locations in each clause and verify that these mentions indeed denote a place. These features describe whether a clause contains a mention of a location and whether it is embedded in a prepositional phrase. The rationale behind these features is that true location mentions are more likely to occur inside prepositional phrases, such as *from Chicago* or *to China.*

In order to meet *Criterion 2* — to select descriptive sentences — the system computes a number of aspect-related features for each clause. These features have been selected to model the characteristics of a clause that help determine its aspectual type. These characteristics include the lexical aspect of the main verb of a clause, tense, temporal expressions, voice and certain properties of the direct object. Each of these properties and its relation to aspect is introduced and explained in **Chapter 5**. **Chapter 5** should be viewed as a motivation behind selecting these particular aspect-related features and not others. The rest of this section concentrates on how the system collects the necessary linguistic information.

Lexical aspect of a verb. As was mentioned in the previous chapter (**Section 5.3.2**), the

Table 6.1. Description of the features in both datasets.

	Fine-grained dataset		Coarse-grained dataset	
Type of features	Number of features	Number of values	Number of features	Number of values
Character-related	10	18	4	6
Aspect-related	14	48	6	15
Location-related	2	4	2	4
Others	3	7	3	4
All	29	77	15	29

knowledge about the lexical aspect of the main verb is very helpful in establishing the aspectual type of the clause. This is why both coarse- and fine-grained representations reflect this information. In order to compute lexical verbal aspect, this work relies on the database of Lexical Conceptual Structures (Dorr and Olsen 1997). It is a manually constructed database that contains a number of syntactic and semantic properties for 4,432 common English verbs. These properties include Levin's class (Levin 1993), sub-categorization frames, *etc*. Dorr and Olsen (1997) define a small set of rules that allows establishing whether a verb is telic, durative and/or dynamic using the information encoded in the LCS database. The system contains an implementation of these rules and computes lexical aspect using them. The fine-grained dataset contains three features with six possible values showing whether the main verb of a clause is durative, dynamic or telic. The coarse-grained dataset contains a single feature with four possible values (the lexical aspect of a verb according to the model in **Table 5.1** in **Chapter 5**).

Many English verbs are polysemous and the LCS database contains more that one entry for them. The system does not perform word sense disambiguation. Instead, it only considers one entry per verb (the first one).

Grammatical tense. The grammatical tense used in a particular clause places a number of constraints on its aspectual type (see **Chapter 5**, **Section 5.3.3** for more details). Since all sentences in each story are parsed using the Connexor Machinese parser, computing tense and grammatical aspect of a clause amounts to retrieving this information from the parse tree (the Connexor parser outputs tense and aspect for all clauses, along with many more morphological and syntactic details). In the fine-grained dataset the information related to tense is expressed using three features with seven possible values (*i.e.,* whether a clause is in present, past or future tense, whether it is progressive and whether it is perfective). In the coarse-grained dataset, this information is expressed using one binary feature: whether a clause is in simple past or present tense.

Temporal expressions. Temporal markers (often referred to as temporal adverbials), such as *usually*, *never*, *suddenly*, *at that moment* and many others, are widely employed in English and often unambiguously signal the aspectual type of a clause. For example:

Example 6.1a. She read a lot tonight.

Example 6.1b. She always read a lot. (Or She used to read a lot.)

Yet, such expressions are not easy to capture automatically. In order to use the information expressed in temporal adverbials, I analyzed the training part of the corpus for presence of such expressions and found 295 occurrences in 10 stories. It appears that this set can be reduced to 95 templates in the following manner. For example, the expressions *during this year*, *during those hours*, *during that hot summer* can all be reduced to a template *during <some_expression>*. Each template is characterized by three features: the type of the temporal expression (location, duration, frequency, enactment) (Harkness 1987); magnitude (year, day, etc.); and plurality (year versus years). The templates are applied using a two-level cascade of regular expressions: at first the system attempts to establish the presence of a unit marking the beginning of a temporal expression (e.g., *during* for the example above). During the next step it attempts to find missing elements, if such elements are needed (e.g., *this year* or that *hot summer*). This is achieved by exhaustively searching a manually composed list of expressions that denote units of time (e.g., *day*, *minute*, names of seasons and months etc.). The fine-grained dataset contains three such features with 14 possible values (type of expression, its magnitude and plurality). The coarse-grained dataset contains one binary feature (whether a clause contains an expression denoting a long period of time). **Appendix D** contains the list of templates for temporal expressions.

Voice. Usually, clauses in passive voice only occur with events (Siegel 1998, p. 51). Both datasets contain one binary feature describing this information.

Properties of direct object. For some verbs, properties of the direct object help determine whether a given clause is stative or dynamic.

Example 6.2a. She wrote a book. (event)

Example 6.2b. She wrote books. (state)

Figure 6.2. Pseudo-code for determining the type of have-clauses based on the WordNet category of direct object (Siegel 1998b.)

```
stateCategories = ['cognition', 'state', 'time', 'artifact',
'attribute', 'entity', 'measure', 'substance', 'relation',
'person', 'group', 'location', 'feeling', 'pronoun', 'ani-
mal']

if parent categories of the hypernym tree contain on
of the stateCategories:
        return True    //stative clause
else:
        return False   //dynamic clause
```

The properties of particular interest include whether the direct object follows a definite or indefinite determiner and whether it is used in a singular or plural form. The fine-grained dataset contains two binary features that describe this information. The coarse-grained dataset contained no such information. It was not included in that dataset because eliminating these features allowed reducing the number of features and cardinality of the set of possible values without removing more informative features.

Several additional features in both datasets describe the overall characteristics of a clause and its parent sentence, such as whether these were affirmative statements, exclamations or questions, their index in the text and a few others. The fine-grained dataset contains four such features with nine possible values and the coarse-grained dataset contained three features with seven values.

6.4. Handling clauses with the verb *have*

The previous chapter mentions that the same verb may form clauses of different aspectual types depending on its context. A special case of a verb that is highly ambiguous with respect to aspect is the verb *have*. The meaning of this verb is strongly influenced by what kind of direct object it takes. That is why determining its aspectual type is a very challenging task. This issue is illustrated in examples *3a-3c*.

Example 6.3a. *She had lunch. (event).*

Example 6.3b. *She had a friend. (state).*

Example 6.3c. She had an accident. (event).

Due to the high degree of ambiguity, the system handles clauses that contain the verb *have* as the main verb in a manner different from all other clauses. This machinery remains the same regardless of what options are used for the granularity of representation and for sentence selection procedures.

In order to handle *have*-clauses, the system contains an implementation of an approach proposed by Siegel (1998b). The solution relies on using WordNet (Fellbaum 1998) and contains a set of rules that determine the aspectual type of a *have*-clause based on the top WordNet category of direct object. For instance, the direct object *lunch* from example *3a* belongs to the category *substance* and, according to rules from (Siegel 1998), the aspectual type of a clause is *event*. The direct object *friend* from example *3b* belongs to the category *person*, so the aspectual type of the clause is *state*. Siegel (1998b) describes an approach that relies on using WordNet 1.6, while I work with a newer version, WordNet 2.0. The structure of this version of WordNet is different from the one for which the pseudo-code was originally designed and the top category of all nouns is of type *entity*. For this reason, the system considers all parent categories in the hypernym tree[36], not only the top one. The pseudo-code is shown in **Figure 6.2**.

The system judges a *have*-clause to be summary worthy if two conditions are fulfilled: the clause contains a mention of one or more important characters and it is a state clause.

6.5. Experiments

6.5.1. Experimental setting

In order to find out how successful the system is in creating summaries of short stories, I conducted a number of experiments. The experimental corpus consists of 47 short stories split into a training set of 27 stories and a test set of 20 stories. The average length of a story in the corpus is 3,333 tokens, 244 sentences or 4.5 letter-sized pages. The corpus contains stories writ-

[36] Except for the top category *entity*.

ten by 17 recognized authors. It was split manually so that its training and test portions contained approximately an equal number of stories by the same writer. I annotated each clause of every story for summary-worthiness and achieved the compression rate of 6%, counted in sentences. This rate constituted the target compression rate in all further experiments.

The training dataset consisted of 10,525 clauses, 506 of which were annotated as summary-worthy and all others – as not summary-worthy. The test dataset contained 7,890 clauses, 406 of them summary-worthy.

Two sets of experiments were conducted; both of them used the training portion of the corpus for fine-tuning the system and adjusting parameters. Once the best settings were identified, the system produced summaries for the test portion of the dataset using them. The first set of experiments consisted of applying a manually designed set of rules that select sentences for inclusion in summaries. These experiments are described in **Section 6.5.2**. The second set of experiments relied on using machine-learning techniques to create summaries. It is described in **Section 6.5.3**. After the completion of the experiments, six judges evaluated the summaries. They were also compared against extractive summaries produced by three people. **Chapter 7** discusses the evaluation procedures in detail and reports the results.

6.5.2. Experiments with manually designed rules

The first classification procedure applies manually designed rules to a clause-level representation of the original stories to produce descriptive summaries. The rules are designed using the same features as those used for machine learning and described in **Section 6.3** and in **Appendix C**.

I created two sets of rules to guide the sentence-classification process: one for the coarse-grained and another for the fine-grained representation. The rules operate at clause level. If a clause is deemed summary-worthy, the complete parent sentence is included in the summary. **Figure 6.3** displays several examples of rules for the fine-grained dataset (a clause is considered to be summary-worthy if a rule returns *True*).

The set of rules for the fine-grained representation has a tree-like structure. It processes the features of a clause and outputs a binary prediction. The rules for the coarse-grained representation function differently. Each clause is assigned a score based on the values of its features. The system then selects 6% of sentences that contain clauses with the highest scores. The scores attributed to the particular feature values were assigned and fine-tuned manually using linguistic knowledge described in **Chapter 5** and in **Section 6.3**. The procedures for the coarse- and the fine-grained datasets differ for two reasons. Assigning and fine-tuning the scores is a more flexible process and it is easier to perform manually. Ideally, I would apply score-based rules to both representations, but assigning and fine-tuning the scores manually for the fine-grained dataset is excessively labour-intensive: there are too many features with too many values.

The rules in both datasets, as well as the set of weights used for the coarse-grained representation, were selected and fine-tuned empirically using the training portion of the corpus as a guide. Once the parameters have been adjusted, the system produced two sets of summaries for the test portion of the corpus (one for each representation).

The pseudo-code for both sets of rules appears in **Appendix E**.

6.5.3. Experiments with machine learning

As an alternative to rule construction, the second set of experiments consisted of applying decision tree induction with C5.0 (Quinlan 1992) to select salient descriptive sentences. I chose C5.0 mainly because of the readability of its output.

The training and test datasets exhibited an almost 1:20 class imbalance (*i.e.*, only 6% of all annotated clauses belonged to the positive class). Because the corpus was rather small, I applied a number of techniques to correct class imbalance in

Figure 6.3. Examples of manually composed rules.

Rule 1
if a clause contains a character mention as subject or object **and** a temporal expression of type enactment (*ever, never, always*)
    return True
Rule 2
if a clause contains a character mention as subject or object **and** a stative verb
    return True
Rule 3
if a clause is in progressive tense
    return False

the training dataset. These techniques included using classification costs, undersampling (randomly removing instances of the majority class), oversampling (randomly duplicating instances of the minority class) and synthetic example generation (Chawlar et al. 2002). Using tenfold cross validation on the training dataset and my original annotations, I selected the best class-imbalance correction techniques for each representation and also fine-tuned learning parameters available in C5.0. These experiments suggested classification costs for the coarse-grained dataset and undersampling for the fine-grained dataset.

In order to see what features were the most informative in each dataset, a small experiment was conducted. I removed one feature at a time from the training set and used the decrease in F-score as a measure of informativeness. The experiment showed that in the coarse-grained dataset the following features were the most informative: the presence of a character in a clause, the difference between the index of the current sentence and the sentence where the character was first mentioned, syntactic function of a character mention, index of the sentence and tense. In the fine-grained dataset the most informative features are the index of the sentence, whether a character mention is a subject, the presence of a character mention in the clause and whether the character mention is a pronoun. After selecting the best parameters on the training dataset, the system produced two sets of summaries for the test dataset. **Appendix C** contains a complete list of features for each dataset along with explanations about how they are computed.

6.6. Conclusion

This chapter contains the description of the system for automatic summarization of short stories. The system functions by selecting sentences from the originals and composing summaries out of them.

The sentences are selected based on two criteria: they must focus on the entities that are central to the story and they must relate background rather than events. The decision as to whether a sentence is summary-worthy or is taken based on a feature-vector representation of its clauses: if one clause is considered to be summary-worthy, then the whole sentence is included into the

Table 6.2. Decrease in F-score caused by removing one feature at a time.

Coarse-grained dataset		Fine-grained dataset	
Feature name	Decrease in f-score, %	Feature name	Decrease in f-score, 5%
char_in_clause	7.3	nbr_of_sent	10.9
nbr_after_first_mention	6.72	char_if_subj	5.56
is_subj_obj	4.53	char_mention	3.72
nbr_of_sent	3.26	char_pronoun	2.53
simple_past_present	2.92	char_if_obj	2.12
is_asser_sent	2.28	clause_type	1.91
past_perfect	2.18	sent_type	1.73
is_assert_clause	1.59	neg	1.45
has_modal	1.52	loc_and_prep	1.4
modified_by_np	1.5	loc_present	1.31
politeness_with_be	1.34	passive	1.23
default_aspect	1.07	durative	1.07
tmp_exp_long_dur	0.61	obj_def	1.06
loc_in_prep	0.61	progr	0.83
loc_present	0.61	char_indef	0.8
		char_modified	0.65
		obj_plur	0.65
		tense	0.65
		dynamic	0.61
		tmp_magn	0.58
		modal	0.55
		tmp_type	0.5
		char_in_sent	0.49
		telic	0.45
		tmp_plur	0.39
		char_if_ind_obj	0.31
		perf	-0.29
		char_is_attr	-0.61

summary. The system offers two options with regard to the clause-level representation: a fine- and a coarse-grained one. It also has two options with respect to the sentence selection process itself: one may use a set of manually designed rules or decision tree induction.

The next chapter explains how the produced summaries were evaluated and presents the results.

Chapter 7. Evaluation of the Automatically Produced Summaries

7.1. Chapter overview

The previous chapter describes the system that produces summaries of short stories. It is the case with most research projects that the results need to be evaluated in order to validate or disprove the proposed approach. The purpose of this chapter is to describe how the automatically produced summaries were evaluated and to interpret the results.

The evaluation procedure consisted of two parts. At first the machine-made summaries were compared with those produced by humans, using several sentence-overlap based measures. This procedure is described in **Section 7.3**. During the next stage several judges read and evaluated a set of summaries by answering factual and subjective questions about them. This stage of the evaluation is explained in **Section 7.4**. **Section 7.5** draws conclusions about the results of the evaluation.

7.2. Overview of the evaluation procedure

The main motivation behind the design of the evaluation procedure was to have easily interpreted, meaningful results, and to keep the cost of labour reasonable. The procedure involved six human subjects who performed two separate tasks.

In Task 1 each subject was asked to read a story and create its summary by selecting 6% of the sentences. The subjects were told that their summaries were to raise expectations about the story, but not to reveal what happens in it.

In Task 2 the subjects made a number of judgments about the summaries before and after reading the original stories. The subjects read a summary similar to the one shown in **Figure 7.1**. Next, they were asked six questions, three of which were factual in nature and three others were

Figure 7.1. Example of a summary produced by the system.

A MATTER OF MEAN ELEVATION. By O. Henry (1862-1910).

On the camino real along the beach the two saddle mules and the four pack mules of Don Señor Johnny Armstrong stood, patiently awaiting the crack of the whip of the arriero, Luis. These articles Don Johnny traded to the interior Indians for the gold dust that they washed from the Andean streams and stored in quills and bags against his coming. It was a profitable business, and Señor Armstrong expected soon to be able to purchase the coffee plantation that he coveted. Armstrong stood on the narrow sidewalk, exchanging garbled Spanish with old Peralto, the rich native merchant who had just charged him four prices for half a gross of pot-metal hatchets, and abridged English with Rucker, the little German who was Consul for the United States. […] Armstrong, waved a good-bye and took his place at the tail of the procession. Armstrong concurred, and they turned again upward toward Tacuzama. […] Peering cautiously inside, he saw, within three feet of him, a woman of marvellous, imposing beauty, clothed in a splendid loose robe of leopard skins. The hut was packed close to the small space in which she stood with the squatting figures of Indians. […] I am an American. If you need assistance tell me how I can render it. […] The woman was worthy of his boldness. Only by a sudden flush of her pale cheek did she acknowledge understanding of his words. […] " I am held a prisoner by these Indians. God knows I need help. […] look, Mr. Armstrong, there is the sea!

subjective. The subjects had to answer these questions using the summary as the only source of information. Subsequently, they read the original story and answered almost the same questions (see **Section 7.4** for details). This process helped understand how informative the summaries were by themselves, without access to the originals, and also to judge whether they were misleading or incomplete.

The experiments were performed on the test portion of the corpus that contained 20 stories. They involved six participants divided into two groups of three people. Group 1 performed Task 1 on stories 1-10 of the testing set and Group 2 performed this task on stories 11-20. During Task 2 Group 1 worked on stories 11-20 and Group 2 – on stories 1-10.

Section 6.5 explains that the system produced four summaries for each story in the test set (the system offers two choices with regard to the clause-level representation (a coarse- and a fine-grained one) and two choices with respect to the sentence selection procedure (machine learning or manually designed rules)). All four versions were compared with human-made summaries using sentence overlap-based measures. However, because the experiments are rather time consum-

ing, it was not possible to evaluate more than one set of summaries using human judgments (Task 2). That is why only the summaries generated using the coarse-grained dataset and manually composed rules were evaluated in Task 2. I selected this version because the differences between this set of summaries and gold-standard summaries are easiest to interpret: decisions based on a set of rules employing a smaller number of parameters are easier to track than those taken using machine learning or more elaborate rules.

On average, the subjects reported that completing both tasks required between 15 and 35 hours of work. Four out of six subjects were native speakers of English. Two others had a near-native and very good levels of English respectively. The participants were given the data in form of files and had four weeks to complete the tasks.

7.3. Creating gold-standard summaries: Task 1

During this task each participant had to create extract-based summaries for 10 different stories. The criteria (making a summary indicative rather than informative) were explained and one example of an annotated story shown. The instructions for these experiments are available at http://www.site.uottawa.ca/~ankazant/instructions.zip.

Table 7.1 presents several measures of agreement between judges within each group and with the annotations produced by me (included in the agreement figures because I created the initial training data and test data for the preliminary experiments).

Table 7.1. Inter-judge agreement.

Statistic	Group 1	Group 2	Average
Cohen (4)	0.50	0.34	0.42
Cohen (3)	0.51	0.34	0.42
PABAK (4)	0.88	0.85	0.87
PABAK (3)	0.89	0.86	0.87
ICC (4)	0.80 (0.78, 0.82)	0.67 (0.64, 0.70)	0.73 (0.71, 0.76)
ICC (3)	0.76 (0.74, 0.80)	0.6 (0.56, 0.64)	0.68 (0.65, 0.72)

The measurement names are displayed in the first column of **Table 7.1**. *Cohen* denotes Cohen's kappa (Cohen 1960). *PABAK* denotes Prevalence and Bias Adjusted Kappa (Bland and Altman 1986). *ICC* denotes Intra-class Correlation Coefficient (Shrout and Fleiss 1979). The numbers *3* and *4* state whether the

statistic is computed only for 3 subjects participating in the evaluation or for 4 subjects (including my annotations).

As can be seen in **Table 7.1**, the agreement statistics are computed for each group separately. This is because the sets of stories that they annotated are disjoint. The column *Average* provides an average of these figures to give a better idea of overall agreement.

Cohen's kappa in its original form can only be computed for a pair of raters. This is why it was computed for each possible pair-wise combination of raters within a group and then the numbers were averaged. The PABAK statistic was computed in the same manner using Cohen's kappa as its basis. ICC is the statistic that measures inter-rater agreement and can be computed for more than 2 judges. It was computed for all 3 or 4 raters at the same time. ICC was computed for a two-way mixed model and measured the average reliability of ratings taken together. The numbers in parentheses are confidence intervals for 99% confidence. (The confidence intervals are not reported for Cohen's kappa and PABAK because these statistics are averages of all pair-wise combinations.)

Computing three different agreement measures was necessary because each of these statistics has its weakness and distorts the results in a different manner. Cohen's kappa is known to be a pessimistic measurement in the presence of a severe class imbalance, as is the case in our setting (Sim and Wright 2005). PABAK is a measure that takes class imbalance into account, but it is too optimistic because it artificially removes class imbalance present in the original setting. ICC has weaknesses similar to Cohen's kappa (sensitivity to class imbalance). Besides, it assumes that the sample of targets to be rated (sentences in our case) is a random sample of targets drawn from a larger population. This is not necessarily the case as the corpus was not compiled randomly.

These three measures, although insufficient individually, provide an adequate understanding of inter-rater agreement in our evaluation. On average, a participant annotated 14.6 sentences per story as summary-worthy. All three judges annotating the same story agreed on an average of 4.4 sentences. That is to say, the average overlap (intersection) between judges in each group is 1.8% out of 6% of summary-worthy sentences.

Figure 7.2. Fragments of summaries produced by 3 annotators for *The Cost of Kindness* by Jerome K Jerome.

Annotator A.
The Rev. Augustus Cracklethorpe would be quitting Wychwood-on-the-Heaththe the following Monday, never to set foot […] in the neighbourhood again. The Rev. Augustus Cracklethorpe, M.A., might possibly have been of service to his Church in, say, […] some mission station far advanced amid the hordes of heathendom. In picturesque little Wychwood-on-the-Heath […] these qualities made only for scandal and disunion. Churchgoers who had not visited St. Jude's for months had promised themselves the luxury of feeling they were listening to the Rev. Augustus Cracklethorpe for the last time. The Rev. Augustus Cracklethorpe had prepared a sermon that for plain speaking and directness was likely to leave an impression.
Annotator B.
The Rev. Augustus Cracklethorpe would be quitting Wychwood-on-the-Heaththe the following Monday, never to set foot […] in the neighbourhood again. The Rev. Augustus Cracklethorpe, M.A., might possibly have been of service to his Church in, say, [..] some mission station far advanced amid the hordes of heathendom. What marred the entire business was the impulsiveness of little Mrs. Pennycoop. Mr. Pennycoop, carried away by his wife's eloquence, added a few halting words of his own. Other ladies felt it their duty to show to Mrs. Pennycoop that she was not the only Christian in Wychwood-on-the-Heath.
Annotator C.
The Rev. Augustus Cracklethorpe would be quitting Wychwood-on-the-Heath the following Monday, never to set foot […] in the neighbourhood again. The Rev. Augustus Cracklethorpe, M.A., might possibly have been of service to his Church in, say, […] some mission station far advanced amid the hordes of heathendom. For the past two years the Rev. Cracklethorpe's parishioners […] had sought to impress upon him, [..] their cordial and daily-increasing dislike of him, both as a parson and a man. The Rev. Augustus Cracklethorpe had prepared a sermon that for plain speaking and directness was likely to leave an impression. The parishioners of St. Jude's, Wychwood-on-the-Heath, had their failings, as we all have. The Rev. Augustus flattered himself that he had not missed out a single one, and was looking forward with pleasurable anticipation to the sensation that his remarks, from his "firstly" to his "sixthly and lastly," were likely to create.

All of these agreement measures and, in fact, all measures based on computing sentence overlap are inherently incomplete where fiction is concerned because any two different sentences are not necessarily “equally different”. **Figure 7.2** exemplifies the matter. It displays segments of summaries produced for the same story by three different annotators. Computing Cohen’s kappa between these fragments gives agreement of 0.521 between annotators A and B and 0.470 between annotators A and C. However, a closer look at these fragments reveals that there are more differences between summaries A and B than between summaries A and C. This is because many

of the sentences in summaries A and C describe the same information (personal qualities of Rev. Cracklethorpe) even though they do not overlap. On the other hand, sentences from summaries A and B are not only distinct; they "talk" about different facts. This problem is not unique to fiction, but in this context it is more acute because literary texts exhibit more redundancy.

Before discussing the results, it is useful to define baselines. As I am not familiar with any comparable summarization experiments, it was impossible to use an existing work for comparison. Therefore, a baseline needed to be defined in different terms. To this end, I have defined two naïve baselines.

Intuitively, when a person wishes to decide whether to read a certain book, she opens it and flips through several pages at the beginning. Imitating this process, I compute a simple lead baseline consisting of the first 6% of the sentences in a story. It is denoted LEAD in **Tables 7.2-7.4**. The second baseline is a slightly modified version of the lead baseline and it consists of the first 6% of the sentences that contain at least one mention of an important character. It is denoted LEAD CHAR in **Tables 7.2-7.4**.

Tables 7.2-7.4 show the results of comparing four different versions of computer-made summaries against gold-standard summaries produced by humans. The tables also display the results of two baseline algorithms. The improvements over the baselines are significant with 99% confidence in all cases.

Combining summaries created by human annotators in different ways results in three distinct types of gold-standard summaries.

Table 7.2. Sentence overlap between computer- and human-made summaries. Majority gold-standard.

Dataset	Prec.	Rec.	F
LEAD	25.09	30.49	27.53
LEAD CHAR	28.14	33.18	30.45
Rules, coarse-grained	34.14	44.39	38.60
Rules, fine-gr.	39.27	50.00	43.99
Machine learning, coarse-gr.	35.55	40.81	38.00
ML, fine-gr.	37.97	50.22	43.22

Table 7.3. Sentence overlap between computer- and human-made summaries. Union gold-standard.

Dataset	Prec.	Rec.	F
LEAD	36.53	17.97	24.09
LEAD CHAR	44.49	21.23	28.75
Rules, coarse-grained	52.41	30.96	38.92
Rules, fine-gr.	56.77	31.22	40.28
Machine learning, coarse-gr.	51.17	23.76	32.47
ML, fine-gr.	55.59	29.76	38.77

The majority gold-standard summary contains all sentences that were selected by at least two judges. It is the most commonly accepted way of creating gold-standard summaries (see **Chapter 2** for a review of standard evaluation practices) and it is best suited to give an overall picture of how similar computer-made summaries are to man-made ones.

Table 7.4. Sentence overlap between computer- and human-made summaries. Intersection gold-standard.

Dataset	Prec.	Rec.	F
LEAD	12.55	37.36	18.78
LEAD CHAR	15.97	46.14	23.73
Rules, coarse-grained	19.66	62.64	29.92
Rules, fine-gr.	23.10	76.92	35.53
Machine learning, coarse-gr.	19.14	53.85	28.24
ML, fine-gr.	21.36	69.23	32.64

The union gold standard is obtained by considering all sentences that were judged summary-worthy by at least one judge. Union summaries provide a more relaxed measurement. Precision for the union gold standard gives one an idea of how many irrelevant sentences a given summary contains (sentences not selected by any of three judges are more likely to prove irrelevant).

The *intersection* summaries are obtained by combining sentences that all three judges deemed to be important. The intersection gold standard is the strictest way to measure the goodness of a summary. Recall for this standard tells one how many of the most important sentences were included in summaries by the system (sentences selected by all three judges are likely to be the most important ones).

The results in **Tables 7.2-7.4** show that the system outperforms both baseline algorithms. They also suggest that the automatically produced summaries bear resemblance to man-made ones. Yet, these results are inconclusive in several ways. As many have remarked on previous occasions (Mani 2001; Radev et al. 2003), sentence overlap does not provide a complete picture of the quality of a summary. First of all, when a summary in question contains sentences that do not appear in any of the reference summaries, one cannot not be sure that those sentences are uninformative or inappropriate for inclusion in a summary. In addition, documents have internal discourse structure and sentences are often inter-dependent. Therefore, even if a summary contains sentences found in one or more reference summaries, it does not always mean that it is sensible to include those sentences in the summary in question.

Yet, despite all these shortcomings sentence overlap is the measure of choice for this work. On the one hand, it has the advantage of being easy to interpret and comprehend. Other possible measures might have included those based on lexical overlap, such as ROUGE (Lin 2004). I chose not to include these measures in this work because their meaning would be difficult to interpret given that fiction is not a genre of text for which those measures were developed. On the other hand, I also decided against deeper approaches, such as the Pyramid method (Nenkova and Passonneau 2004), factoids (Van Halteren and Teufel 2003) and relative utility (Radev and Tam 2003). The reason for this decision is practical: these approaches have an unfortunate disadvantage of being considerably more labour-intensive than the measures based on sentence overlap.

Looking at **Tables 7.2-7.4** one may wonder about the causes of errors. I have performed manual analysis of the summaries (all four sets) to discover the most common causes. This work incorporates many different tools and algorithms, such as GATE gazetteer, the anaphora resolution algorithms, the Connexor Machinese Syntax parser, LCS database, to name just a few. Each of these tools occasionally makes mistakes, which propagate and influence the final result. It seems that the most harmful errors occur when the gazetteer and the NE recognizer fail to identify the main characters in the story. This is not common and happens mostly when names are of non-English origin or when the main characters are animals. The errors of the anaphora resolution module are more common but seem to be less harmful: even when the system incorrectly links two character mentions, the error is not crucial as long both characters involved in the mistake are important to the story. However, the system does not replace pronouns with the antecedent names in the final version of the summary. If it did, the mistakes of the anaphora resolution module would be more severe and would worsen the quality of the summaries.

The errors of the parser do not seem to influence the performance of the anaphora resolution module, yet they affect the process of selecting descriptive sentences. The Connexor parser performs considerably better than others commonly used parsers[37], but it makes mistakes nevertheless: some of the stories in the corpus contain sentences than span many lines and very elaborate

[37] Before selecting the parser I tried the Link parser, the ApplePie parser, Xerox Incremental Parser and Collins parser. All of the above seemed to have to many difficulties in handling multi-clausal sentences, which are very frequent in fiction. The Connexor parser handles multi-clausal sentences far better.

language (*e.g.*, stories by Jerome K. Jerome). The parser occasionally splits such sentences and they may later appear in the summaries as fragments and severely worsen the readability of the summary. The sentence selection module also tends to make mistakes when processing sentences or clauses in imperative (e.g., *Look!* or *Do not do that!)* and such sentences can be encountered in the summaries because the system judges them to be stative (*e.g.,* the last sentence of the summary in **Figure 7.1**.)

The current system handles direct speech and dialogues in the same manner as all other clauses. That is why sentences containing direct speech often cause errors (imperative clauses are a special case of this issue). The punctuation and spelling in dialogues often pose a problem for the parser and, therefore, for the rest of the system. Even when stative clauses focusing on main entities are correctly identified, it may be hard for a reader to infer to whom the quote belongs. Such sentences are confusing and decrease the readability of the summaries.

A separate problem arises when a story exhibits unusual structure, such as embedded narrative (*i.e.,* a story involves someone telling a story) or non-linear narrative (*i.e.,* the story starts at a certain point in time and then goes back to provide necessary background information). This is especially problematic for the machine learning approaches because they rely heavily on the position of sentences in the text. In order to avoid this problem some sort of discourse-level processing needs to be performed.

The analysis of the causes of errors in the summaries suggests that summaries may be significantly improved by improving the performance of out-of-the-box tools involved in producing them. Yet, some of the errors that are found in the summaries cannot be overcome no matter how much the tools are improved. Even at its best, the current approach offers no solution to making use of the figurative language and humor in fiction; nor can it capture states and events that are not explicitly expressed in the text. This suggests that using deeper, more semantically motivated technologies may be necessary.

7.4. Human judgment of computer-made summaries: Task 2

In Task 2, each subject evaluated 10 automatically produced summaries. The task had two parts. At first, the participants were required to read a summary and answer six questions about it. The subjects had to answer the questions using the summary as the only source of information. Next, they read the original story and answered another six questions – this time using the knowledge obtained from the original. The questions asked after reading the summary and after reading the story were the same with one exception, which is explained below. Three of the questions were factual and three others called for a subjective judgment. The subjects were asked not to correct the first set of answers after reading the complete story.

Table 7.5 displays the factual questions along with the resulting answers. The first column shows numerical ids of the questions (*e.g., Q1*). The ids Q1-Q3 correspond to the questions that were answered using the summary only; questions Q7-Q9 were answered using the complete story. The factual questions were as follows: a) list up to three main characters of the story in the order of importance b) state where the story takes place and c) state when the story takes place. The subjects had to answer the questions about main characters (Q1 and Q7) and about location (Q2 and Q8) in their own words and the answers were rated on the scale of -1 to 3. A score of 3 meant that the answer was complete and correct, 2 – slightly incomplete, 1 – very incomplete, 0 – a subject could not find the answer in the text and -1 if the person answered incorrectly. The questions asking to identify the time frame of a story (Q3 and Q7) were multiple-choice questions where participants had to select the century when a story took place. The answers to these questions were rated on the binary scale (1 if the answer was correct and 0 if it was not or when a

Table 7.5. Answers to factual questions.

Id	Question	After summary only		After reading the original	
		Mean	Std. dev	Mean	Std. dev.
Q1, Q7	Please list up to 3 main characters in this story, in the order of importance (scale: -1 to 3)	2.28	0.64	2.78	0.45
Q2, Q8	State where this story takes place. Be as specific as possible (scale: -1 to 3)	1.78	1.35	2.60	0.91
Q3, Q9	Select a time period when this story takes place.(scale: 0 or 1)	0.53	0.50	0.70	0.46

subject could not infer time from the text).

The subjective questions and the results appear in **Table 7.6**. The questions Q4 – Q6 were answered after reading the summary, while Q10-Q12 – after reading the complete story. The questions requiring the participants to pronounce a subjective judgment were as follows: a) how readable do you find this summary (Q4 in **Table 7.6**) b) how much irrelevant information does this summary contain (Q5 and Q10 in **Table 7.6**) c) how complete is the summary (Q11 in **Table 7.6**) and d) how helpful was this summary in deciding whether you would like to read the story (Q6 and Q12 in **Table 7.6**). The subjective questions asked after reading the summary and after reading the original remained unchanged with one exception – Q4 (how readable do you find the summary) was replaced by question Q11 (how complete is the summary). All subjective questions were multiple-choice questions where a judge needed to select a score from 1 to 6, with 1 indicating a strong negative property and 6 indicating a strong positive property.

The evaluation procedure included both types of questions (*i.e.,* factual and subjective) because each type provided insights into different qualities of the summaries. The subjective questions were intended to do just what the name suggests, that is subjective evaluation. The motivation behind them was that they would reveal how useful or useless the judges found the summaries on the accounts in question. The factual questions were meant to provide a different type of information: how informative the summaries were. These questions also have the advantage of being more objective: a person may be biased in favor or against the summary due to how much she likes or dislikes the story. The bias may affect the answers to the subjective questions but it is unlikely to influence how a person answers a question such as where the story takes place.

The results displayed in **Tables 7.5** and **7.6** suggest that the subjects can answer simple ques-

Table 7.6. Answers to subjective questions.

Id	Question (scale: 1 to 6)	After summary only		After reading the original	
		Mean	Std. dev	Mean	Std. dev.
Q4	How readable do you find this summary?	4.43	1.39	N/A	N/A
Q5, Q10	How much irrelevant information does this summary contain?	4.27	1.41	4.51	1.16
Q11	How complete is the summary?	N/A	N/A	4.53	1.25
Q6, Q12	How helpful was this summary for deciding whether you would like to read the story or not?	4.52	1.37	4.6	1.21

tions based on the summaries alone. They also seem to indicate that the subjects found the summaries quite helpful in achieving the original objective. It is interesting to note that even after reading complete stories the subjects are not always capable of answering the factual questions with perfect precision.

7.5. Conclusions

This chapter describes how the automatically produced summaries of the short stories have been evaluated. The procedure involved six human subjects and included applying both extrinsic and intrinsic measures to evaluate each summary. The results suggest that the automatically produced summaries resemble man-made ones. They also indicate that the subjects are able to answer simple questions about the complete story using the summary as the only source of information. The subjects also found the summaries helpful for deciding whether they wanted to read the original.

The error analysis indicates that the summaries may be significantly improved by reducing the error rates of the intermediate tools (*e.g.,* named-entity recognition). It also suggests that in order to achieve understanding of the stories and to produce flexible summaries, deeper semantic analysis may be required.

Chapter 8. Conclusions and Future Work

8.1. Contributions

In **Chapter 1** I defined the goal of this work as producing summaries of short stories with a specific objective of helping a reader make informed decisions with respect to the complete story. **Chapters 4-6** described a system that does just that – it produces indicative extractive summaries of short stories without revealing the plot. The system relies on using a number of surface and syntactic cues about the original documents. These cues can be divided into two distinct categories: information about the important entities in the texts (*i.e.,* locations and characters) and information about the aspectual type of clauses. The system creates summaries of short stories by selecting sentences in the original documents that focus on main characters or locations and that are descriptive (rather than event sentences). **Chapter 7** explained how the summaries were evaluated. The results of the evaluation suggest that the system produces summaries that resemble those created by humans and that people find them useful for forming adequate expectations about the stories. Therefore, the objective (helping a reader decide whether she wants to read the original story) has been achieved, if only with partial success.

The most important contribution of this work is the following: it shows that reasonably successful summarization of short fiction is feasible using basic state-of-the-art technologies in the area of Natural Language Processing. The summarization system described in **Chapters 4**, **5** and **6** relies on a number of tools and algorithms proposed by other researchers. Most of these tools are publicly available and require no extraordinary resources[38]. Unfortunately, I had no access to the tools that would perform deep semantic analysis. If one were to approach this task armed with such tools, with more precise named-entity recognition and anaphora resolution, with near-perfect tools for analyzing syntax and properties of the verbs, one may expect the results far exceeding those achieved in the course of this work.

[38] The only exception is the Connexor Machinese Syntax parser.

A separate contribution of this work is that it suggested several characteristics of short stories as a genre that may be helpful for automatic summarization. The foremost of these findings is the fact that characters are central to this genre of fiction. This may seem as the statement of the obvious but to the best of my knowledge this fact had never been explicitly used for automatic summarization. The work of Wendy Lehnert (1982) also confirms this finding, although implicitly: her framework for summarizing narratives relies on a graph representing the mental states of the characters.

In addition, one may notice that the position of sentences in the text plays an important role when deciding whether a sentence is summary-worthy, even though the position is not as crucial as it is in structured documents (*i.e.,* newspaper articles). Several findings support this idea: first of all, the position in text is one of the most important features in the decision trees induced by C5.0 (see **Chapter 6**). This is so in all four combinations of the clause representation and sentence selection procedures. In addition, when looking at the results of comparing machine-made summaries against men-made ones, one may notice that the lead baselines' performance is not far worse than the performance of the summarization program. The system consistently outperforms the baselines by a statistically significant, yet not very wide margin.

8.2. Shortcomings

This work is exploratory and as such it has a number of omissions and weaknesses. Perhaps the main shortcoming of the approach presented in this dissertation is the fact that it is not capable of handling either the plot or events. Although this is in line with the original objective, having the possibility of identify and summarize the elements of the plot in the stories would allow one to build customized flexible summaries suitable for multiple purposes. This work does not address this issue, which certainly merits investigation.

One may also call into question the limited scope of the summaries – they only concentrate on the main characters and locations and do not explicitly identify any other types of entities. I expected (and the expectation has been realized to a large extent) that since the summaries are extractive, they would include elements that cannot be found by looking at surface indicators (*e.g.,*

information about the timeframe of the story, its style, language, *etc.*) and that this implicit information would invoke a corresponding schema in the reader's mind. Yet, if the system were able to explicitly identify elements other than locations and characters, it would certainly improve the quality of the summaries.

In addition, the summarization system only produces extractive summaries. In the long run, extractive summarization of fiction is unlikely to prove sufficient. On the other hand, producing the abstracts of the works of fiction may prove complicated: because of ethical considerations[39], ungrammatical or incoherent sentences are less acceptable than in some other types of documents.

8.3. Future work

In the future, this line of research may evolve in several directions. A number of technologies may prove useful for improving the quality of the summaries of short stories and/or expanding their scope.

The approach presented in this dissertation only relies on syntax and surface information about the sentences in the stories. However, the structure of a narrative extends beyond intra-sentential one. Establishing discourse-level structure of such documents would provide one with a wealth of information that can be leveraged for summarization. It is unclear whether any of the existing discourse structure theories can be directly applied to fiction because few of them are capable of dealing with literary devices. In addition, the discourse structure of a narrative is tightly connected with its temporal structure: in order or identify flashbacks, descriptions of future plans and events happening simultaneously, one needs to be able to establish the time-line of the narration. It is also possible that in order to summarize the plot of the story, one will need to be able to identify events.

[39] Fiction, at least most of it, is a form of art. It is not quite clear how ethical it is to take a written work of art and automatically generate an abstract of it. This is especially so because the state-of-the-art Language Generation technologies still fall short of producing coherent and easily readable text.

It may also be helpful to pre-categorize the stories into predefined categories prior to summarization[40]. The works of Schmidt (2001) and Booker (2004) suggest that short stories may be categorized according to their main characters, their plot and a few other characteristics. It is unlikely that building template-like summaries will be a successful way to summarize fiction, but the knowledge about the type of a story may help identify useful elements in the text.

Regardless of the type of the summaries produced, their scope and purpose, one direction for the future research is clear: the study of summarizing fiction needs to be an interdisciplinary one. It must include the knowledge about the genre from the literary studies community, it is likely to require linguistic knowledge about this type of texts and the savoir-faire of Computer Science in order to produce informative, cohesive and useful summaries.

[40] I would like to thank Peter Turney for suggesting and discussing this idea.

References

Aristotle. *The metaphysics : books I-IX / Aristotle; with an English translation by Hugh Tredennick*. Cambridge, MA, Harvard University Press, 1980.

Regina Barzilay and Kathleen McKeown. 2005. Sentence Fusion for Multidocument News Summarization, In *Computational Linguistics*, 31(3):297-329.

H.E. Bates.1972. *The Modern Short Story*. Evensford Productions Ltd.

Adam Berger, Stephen Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. In *Computational Linguistics*, 22(1): 39-71.

Robert I. Binnick. 1991. *Time and the Verb: A Guide to Tense and Aspect*. Oxford University Press, New York Oxford.

J. Martin Bland and Douglas G. Altman. 1986. Statistical methods for assessing agreement between two methods of clinical measurement. In *Lancet*. 1986; 1(8476):307-310.

Christopher Booker. 2004. *The Seven Basic Plots. Why we tell stories*. Continuum, London.

Chris Buckley, James Allan and Gerard Salton. 1993. Automatic Routing and Ad-hoc Retrieval Using SMART: TREC 2. In *TREC 1993*, 45-56.

Razvan Bunescu. 2003. Associative Anaphora Resolution: A Web-Based Approach. In *Proceedings of the EACL 2003 Workshop on The Computational Treatment of Anaphora*.

Tomas By. 2002. *Tears in the Rain*. Ph.D. thesis, University of Sheffield.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. In *Computational Linguistics*, 22(2): 249-254.

Eugene Charniak. 1972. *Toward a Model of Children's Story Comprehension*. Ph.D. thesis, Massachusetts Institute of Technology.

Nitesh V. Chawlar, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over–sampling techniques. In *Journal of Artificial Intelligence Research* (16): 321-357.

Nello Cristianini. and John. Shawe-Taylor. (2000) An introduction to Support Vector Machines. Cambridge University Press.

J. Cohen, 1960. A coefficient of agreement for nominal scales. In *Educational and Psychological Measurement*, (20): 37-46.

Bernard Comrie. 1976. *Aspect*. Cambridge University Press.

Terry Copeck, Natalie Japkowicz and Stan Szpakowicz. 2002. Text Summarization as Controlled Search. In *Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002 Calgary, Canada, May 27-29, 2002*. Lecture Notes in Artificial Intelligence 2338, Springer Verlag, 268-280.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia*.

Teun van Dijk. 1979. Recalling and Summarizing Complex Discourse. In W. Burchart and K. Hulker (eds.), *Text Processing*, 49-93. Berlin: Walter de Gruyter.

Marin Dimitrov. 2002. *A Light-Weight Approach to Coreference Resolution for Named Entities in Text*. M.Sc. Thesis, University of Sofia, Bulgaria.

Bonnie J. Dorr and Mari Broman Olsen. 1997. *Deriving verbal and compositional lexical aspect for NLP applications.* In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics*, 151–158.

David Dowty. 1979. *Word Meaning and Montague Grammar*. D. Reidel Publishing Company, Dordrecht.

H.P. Edmundson. 1969. New Methods in Automatic Extracting. In *Journal of the Association for Computing Machinery,* 16(2): 264-285.

Noemie Elhadad, Min-Yen Kan, Judith Klavans, and Kathleen McKeown. 2005. Customization in a Unified Framework for Summarizing Medical Literature. In *Journal of Artificial Intelligence in Medicine*, 33(2): 179-198.

Brigitte Endres-Niggemeyer. 1998. Summarizing Information. Berlin, Springer.

James Fan, Bruce Porter and Ken Barker. 2005. Indirect Anaphora Resolution as Semantic Path Search. In *Proceedings of the Third International Conference on Knowledge Capture (K-CAP 2005)*, 153-160.

Atefeh Farzindar and Guy Lapalme. 2004. Legal text summarization by exploration of the thematic structures and argumentative roles. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 27-38, Barcelona, Spain, July 2004.

Fellbaum, Christiane, ed., 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-Based Extractive Summarization. In the *Proceedings of ACL Workshop on Summarization, Barcelona, Spain, July 2004*, 104-111.

Niyu Ge, John Hale and Eugene Charniak. 1998. A Statistical Approach to Anaphora Resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora, 1998,* 161-170.

Alexander Gelbukh and Grigori Sidorov. 1999. On Indirect Anaphora Resolution. In *Proceedings of PACLING-99, Pacific Association for Computational Linguistics*, 181-190.

George W. Gerwig. 1971. *The Art of the Short Story*. Folcroft Library Editions.

Jefferson Graham. 2004. Google's library plan a 'huge help.' In *USA Today*, 12/04/2004.

Barbara Grosz and Candace Sidner. 1986. Attention, intention, and the structure of discourse. In *Computational* Linguistics, 12(3): 175-204.

M.A.K Halliday and Ruqaiya Hasan. 1996. Cohesion in English. London, Longman.

Janet Harkness. 1987. Time Adverbials in English and Reference Time. Alfred Schopf (ed.), Essays on Tensing in English, Vol. I: Reference Time, Tense and Adverbs, 71-110. Tübingen: Max Niemeyer.

Marty Hearst. 1997. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages, In *Computational Linguistics*, 23 (1): 33-64.

Jerry Hobbs. 1978. Pronoun Resolution. In *Lingua*, (44):339-352.

Jerry Hobbs. 1985. On the Coherence and Structure of Discourse, Report No. CSLI-85-37, Center for the Study of Language and Information, Stanford University.

Rodney Huddleston and Geoffrey Pullum. 2002. The Cambridge Grammar of the English Language Usage. Cambridge University Press.

Hongyan Jing, Regina Barzilay, Kathleen McKeown and Michael Elhadad. 1998. Summarization Evaluation Methods: Experiments and Analysis. In *Proceedings of the AAAI Spring Symposyum on Intelligent Text Summarization*, 60-68.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *the Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99),* 129-136.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila. 1995. Constraint grammar: a language-independent system for parsing unrestricted text. Berlin/New York, Mouton de Gruyter.

Christopher Kennedy and Branimir Boguraev. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th Conference on Computational Linguistics*, 113-118.

Jon M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. In *Journal of the Association for Computing Machinery*, 46(5): 604-632.

Damon Knight. 1981. *Creating Short Fiction*. Writer's Digest Books.

Anna Korhonen. 2000. Using Semantically Motivated Estimates to Help Subcategorization Acquisition. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, Hong Kong*, 216-223.

Shalom Lappin and Herbert Leass. 1994. An algorithm for Pronominal Anaphora Resolution. In *Computational Linguistics*, 20(4): 535-561.

Wendy Lehnert. 1982. Plot Units: A Narrative Summarization Strategy. In W. Lehnert and M. Ringle (eds.). *Strategies for Natural Language Processing*, 223-244. Erlbaum, Hillsdale, NJ.

Beth Levin. 1993. *English Verb Classes and Alternations*. The University of Chicago Press.

Wenjie Li, Mingli Wu, Qin Lu, Wei Xu and Chunfa Yuan. 2006. Extractive Summarization using Inter- and Intra- Event Relevance. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 369-376.

Elizabeth Liddy. 1991. The discourse-level structure of empirical abstracts: an exploratory study. In *Information Processing and Management: an International Journal*, 27(1): 55-81.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74-81.

Chin-Yew Lin and Eduard Hovy. Identifying Topics by Position. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP97)*, 283-290.

Inderjeet Mani. 2001. *Automatic Summarization.* John Benjamins B.V.

Inderjeet Mani and Eric Bloedorn. 1997. Multi-Document Summarization by Graph Search and Matching. *In AAAI/IAAI 1997*, 622-628.

Inderjeet Mani and Eric Bloedorn. 1999. Summarizing Similarities and Differences Among Related Documents. In *Information Retrieval*, 1(1), 35-67.

William Mann and Sandra Thompson. 1987. *Rhetorical structure theory: A theory of text organization*. NIST Report ISI/RS-87-190 NTIS: ADA 183038.

Daniel Marcu. 1997. From Discourse Structures to Text Summaries. *The Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization,* 82-88.

Daniel Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *Proceedings of the Sixth Workshop on Very Large Corpora,* 206-215.

Daniel Marcu. 2000. The Theory and Practice of Discourse Parsing and Summarization. The MIT Press.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. In *Computational Linguistics*, 19:313-330.

Brander Matthews. 1888. *The Philosophy of the Short-Story*. In Charles May (ed.), *The New Short Story Theories*, Ohio University Press, 1994, 73-80.

Charles May. 1994. *Chekhov and the Modern Short Story*. In Charles May (ed.), *The New Short Story Theories*, Ohio University Press, 1994, 73-80.

M. C. McCord. 1980. Slot Grammars. In *Computational Linguistics*, 6:31-43.

Ryan McDonald. Discriminative Sentence Compression with Soft Syntactic Evidence. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento Italy, 297-304.

Paola Merlo, Suzanne Stevenson, Vivian Tsang and Gianluca Allaria. 2002. *A Multilingual Paradigm for Automatic Verb Classification*. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 207-214.

Rada Mihalcea and Paul Tarau. 2005. An Language Independent Algorithm for Single and Multiple Document Summarization. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP), Korea, October 2005*, 19-24.

Ruslan Mitkov. 2002. Anaphora Resolution. Pearson Education.

Natalia Modjeska, Katja Markert, and Malvina Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 176-183.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating Content Selection in Summarization: the Pyramid Method. In Proceedings of NAACL-HLT 2004, 145-152.

Laurence Page, Sergey Brin, Rajeev Motwani and Terry Winograd. 1998. The PageRank Citation Ranking: Bring Order to the Web. Technical Report, Stanford University.

Allan H. Pasco. 1991. On Defining Short Stories. In *New Literary History,* 22(4):407-422.

Rebecca Passonneau, Ani Nenkova, Kathleen McKeown, Sergey Sigelman, 2005. Applying the Pyramid Method in DUC 2005. *In Proceedings of the 2005 Workshop of the Document Understanding Conference (DUC)*.

Ted Pedersen, Siddharth Patwardhan and Jason Michelizzi. 2004. WordNet::Similarity – Measuring the Relatedness of Concepts. In *Proceedings of AAAI*, 25-29.

Allan Edgar Poe. 1842. Poe on Short Fiction. In Charles May (ed.), *The New Short Story Theories*, Ohio University Press, 1994, pp. 59-72.

Massimo Poesio and Renata Vieira. 2000 . An Empirically Based System for Processing Definite Descriptions. In *Computational Linguistics*, 26(4): 525-579.

Livia Polanyi. 1989.Telling the American Story. MIT Press.

J. Ross Quinlan. 1992. C4.5: Programs for Machine Learning. Morgan Kaufmann Pub., San Mateo, CA.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech. and Jan Svartvik. 1985. A Comprehensive Grammar of the English Language. Longman, London.

Dragomir R. Radev, Vasileios Hatzivassiloglou, and Kathleen R. McKeown. 1999. A description of the CIDR system as used for TDT-2. In *DARPA Broadcast News Workshop*.

Dragomir Radev and Daniel Tam. 2003. Summarization evaluation using relative utility. *Proceedings of the twelfth international conference on Information and knowledge management*, 508-511.

Dragomir Radev, Simone Teufel, Horacio Saggion, and W. Lam. 2003. Evaluation challenges in large-scale multi-document summarization. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 375-383.

Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, Daniel Tam. 2004. Centroid-based summarization of multiple documents. In *Information Processing and Management*, 40(6): 919-938.

G. Rath, A. Resnik and R. Savage. 1961. The formation of abstracts by selection of sentences. Part 1: sentence selection by man and machines. In *American Documentation*, 12(2): 139-141.

Gerard Salton and Chris Buckley. 1988 Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, 24(5): 513–523.

Gerard Salton, Amit Singhal, Mandar Mitra, Chris and Buckley. 1997. Automatic Text Structuring and Summarization. In *Information Processing and Management*, 33(2): 193-208.

Victoria Lynn Schmidt. 2001. *45 Master Characters.* Writer's Digest Books.

P. Shrout and J. Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. In *Psychological Bulletin*, 86:420–428

Julius Sim and Chris Wright. 2005. The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements. In *Physical Therapy*, 2005(85-3): 257-268.

Eric V. Siegel. 1998. Linguistic Indicators for Language Understanding: Using machine learning methods to combine corpus-based indicators for aspectual classification of clauses. Ph.D. Dissertation. Columbia University.

Eric V. Siegel. 1998b. Disambiguating Verbs with the WordNet Category of the Direct Object. In *Usage of WordNet in Natural Language Processing Systems: Proceedings of the Conference, Université de Montréal, August, 1998*, 9-19.

Sidney Siegel and John Castellan. 1988. *Non parametric statistics for the behavioral sciences*. McGraw-Hill, New York.

E.F. Skorohodko. 1972. Adaptive Method of Automatic Abstracting and Indexing. In Information Processing 71: Proceedings of the IFIP Congress 71, C.V. Friedman (ed.), 1179-1182. Amsterdam, North Holland.

Wee Men Soon, Hwee Tou Ng and Daniel Chung Yong Lim. 2003. A Machine Learning Approach to Coreference Resolution of Noun Phrases. In *Computational Linguistics,* 27(4):521:544.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, 64-71.

Simone Teufel and Marc Moens. 2002. Summarizing Scientific Articles - Experiments with Relevance and Rhetorical Status. In *Computational Linguistics*, 28 (4): 409-445.

Gian Lorenzo Thione, Martin van den Berg, LiviaPolanyi and Christopher Culy. 2004. Hybrid Text Summarization: Combining external relevance measures with Structural Analysis. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 51-55.

Peter Turney. 2002. Learning algorithms for keyphrase extraction. In *Information Retrieval* 2(4):303-336.

Hans Van Halteren and Simone Teufel. 2003: Examining the consensus between human summaries: initial experiments with factoid analysis. In *HLT/NAACL-2003 Workshop on Automatic Summarization*, 57-64.

Zeno Vendler. 1967. *Linguistics in Philosophy*. Cornell University Press, 97- 145.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin and Craig G. Nevill Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of ACM DL-99*, 254-256.

Klaus Zechner. 2002. Automatic Summarization of Open-Domain Multiparty Dialogues in Diverse Genres. In *Computational Linguistics,* 28(4):447-485.

Liang Zhou and Eduard Hovy. Digesting Virtual "Geek" Culture: The Summarization of Technical Internet Relay Chats. In *Proceedings of Association for Computational Linguistics (ACL-2005), Ann Arbor, MI*, 298-305.

Appendix A. Stories in the corpus.

The training set:

Vanka by Anton Chekov.
Gooseberries by Anton Chekov.
The Christmas Tree and the Wedding by Fiodor Dostoevsky.
The Frog and the Puddle by Edna Ferber.
The Kitchen Side of the Door by Edna Ferber.
Araby by James Joyce.
How the Camel Got His Hump by Rudyard Kipling.
How the Whale Got His Throat by Rudyard Kipling.
The White Man's Way by Jack London.
The Lady's Maid by Katherine Mansfield.
Miss Brill by Katherine Mansfield.
The Garden Party by Katherine Mansfield.
An Artifice by Guy de Maupassant.
The Duel by Guy de Maupassant.
Regret by Guy de Maupassant.
The Umbrella by Guy de Maupassant.
The Memoirs of a Yellow Dog by O. Henry.
The Gift of the Magi by O. Henry.
God Sees the Truth but Waits by Leo Tolstoy.
A Dog's Tale by Mark Twain.
The Last of the Culkinses by Artemus Ward.
The Nightingale and the Rose by Oscar Wilde.

The test set:

Anyuta by Anton Chekov.

The Darling by Anton Chekov.

Polinka by Anton Chekov.

One of the Old Girls by Edna Ferber.

The Man Who Came Back by Edna Ferber.

What She Wore by Edna Ferber.

The Outcasts of the Poker Flat by Bret Harte.

The Cost of Kindness by Jerome K. Jerome.

That Spot by Jack London.

Her First Ball by Katherine Mansfield.

An Ideal Family by Katherine Mansfield.

The Necklace by Guy de Maupassant.

That Costly Ride by Guy de Maupassant.

A Call Loan by O. Henry.

The Brief Debut of Tildy by O. Henry.

The Indian Summer of Dry Valley Johnson.

The Mammon and the Archer by O. Henry.

The Matter of Mean Elevation by O. Henry.

The Red Chief by O. Henry.

Appendix B. Lists of nouns added to GATE gazetteer.

animals.lst	person_nouns.lst	person_nouns_female.lst	per-son_nouns_male.lst
monkey	client	bride	boy
deer	person	Bride	brother
hare	companion	daughter	chap
dog	friend	daughter-in-law	father
Dog	child	Duchess	father-in-law
Flea	intruder	girl	host
Grasshopper	woodcutter	girlfriend	husband
Leap-frog	vicar	grand-mother	gentleman
housedog	neighbor	grandmother	grand-father
Cock	neighbour	mistress	grandfather
Stork	parishioner	Mistress	guy
Wind	artist	mother	magus
Dew	baby	Mother	man
Mouse	Baby	mother-in-law	Man
nightingale	doctor	sister	prophet
Catacean	orphan	stepdaughter	samurai
'Stute Fish	Orphan	step-daughter	Samurai
horse	partner	stepmother	soldier
Horse	patient	step-mother	son
Lizard	manager	witch	son-in-law
Butterfly		woman	stepfather
Daisy		princess	step-father
Camel		Princess	uncle
Ox		Queen	wizard
		queen	councilor
		wife	king
		Wife	prince
		Dryad	Prince
		dryad	King
		granny	Host
		Granny	Husband
		Nanny	seaman
		nanny	Seaman
		maiden	postman
		Maiden	papa
		Empress	Papa
		empress	Emperor
		concubine	emperor
		Concubine	swineherd

		widow Widow hostess Hostess aunt Aunt	Swineherd poet Poet Patriot page Grandpapa lad Lad bridegroom Bridegroom kid Kid mountaineer master mariner student Student officer Officer Djinn djinn fellow Fellow

Appendix C. Features used in the coarse- and the fine-grained clause representations

The following appendix lists features that are computed to represent a clause in the fine-grained dataset (Table C.1) and in the coarse-grained dataset (Table C.2). Prior to constructing feature vectors, the stories are parsed with Connexor Machinese Parser. All syntactic information is computed on the basis of the parser output. The column Category shows whether a feature is character-related (C), a location-related (L), aspect-related (A) or other (O).

Table C.1. Features representing a clause in the fine-grained dataset.

Name	Category	Possible values	Description	Default value
char_if_ind_obj	C	*yes, no*	*yes* if the clause contains a mention of a character and its grammatical function is indirect object	*no*
char_if_obj	C	*yes, no*	*yes* if the clause contains a mention of a character and its grammatical function is direct object	*no*
char_if_subj	C	*yes, no*	*yes* if the clause contains a mention of a character and its grammatical function is subject	*no*
char_in_sent	C	*yes, no*	*yes* if the parent sentence contains a mention of a character	*no*
char_indef	C	*def, indef*	*def* if the clause contains a mention of a character and a) it is a proper name or b) it is modified by a definite determiner or a pronoun; *indef* if the mention is modified by an indefinite determiner.	*n/a*
char_is_attr	C	*yes, no*	*yes* if the mention of a character is in the genitive case	*n/a*
char_mention	C	*yes, no*	*yes* if the clause contains a mention of a character	*no*
char_modified	C	*yes, no*	*yes* if the mention of a character is modified by a noun phrase	*n/a*
char_pronoun	C	*1st, 3rd*	*1st* if the clause contains a pronominal mention of a character and it is in 1st person (e.g., *I*); *3rd* if the pronominal mention is in 3rd person (e.g., *he*)	*n/a*
nbr_after_first_mention	C	*continuous*	an integer that reflects the difference between the index of the current sentence and the sentence where the character is first mentioned (it is only defined for clauses containing mentions of characters)	*-1*
loc_in_prep	L	*yes, no*	*yes* if the clause contains a mention of a location and is embedded in a prepositional clause	*no*
loc_present	L	*yes, no*	*yes* if the clause contains a mention of a location	*no*
durative	A	*yes, no*	*yes* if the main verb of the clause is durative; this information is computed using LCS	*no*
dynamic	A	*yes, no*	*yes* if the main verb of the clause is dynamic; this information is computed using LCS	*no*
modal	A	*can, could,*	a modal verb from the list, if it appears in the clause	*n/a*

		shall, should, would, must, may, might, dare, need, will, ought, canst		
neg	A	*yes, no*	*yes* if the main verb of the clause is negated	*no*
obj_def	A	*yes, no*	*no* if the direct object of the main verb is modified by an indefinite determiner; *yes* in all other cases where a direct object is present	*n/a*
obj_plur	A	*yes, no*	*yes* if the direct object of the verb is in plural; *no* in all other cases where a direct object is present	*n/a*
passive	A	*yes, no*	*yes* if the clause is realized in passive voice	*no*
perf	A	*yes, no*	*yes* if the clause is realized in a perfect tense	*no*
progr	A	*yes, no*	*yes* if the clause is realized in a progressive tense	*no*
telic	A	*yes, no*	*yes* if the main verb of the clause is telic; this information is computed using LCS	*no*
tense	A	*past, present, future*	the tense used in the clause	*n/a*
tmp_magn	A	*min, hour, day, week, month, year, year_plus*	the magnitude of the core temporal unit in the expression (defined for clauses containing temporal expressions and assigned using a set of manually designed templates): *min* if the core unit denotes a period of no more than a minute (e.g., *in a few seconds, that moment*); *hour* if it denotes a period of no more than an hour (e.g., *during those hours, at 10 am*); the values *day* through *year* are assigned analogously, and *year_plus* denotes periods longer than a year (e.g., *for decades*)	*n/a*
tmp_plur	A	*yes, no*	*yes* if the core temporal unit in the expression is in plural (e.g., *during those years*), no – if it is singular (e.g., *that day*); defined for clauses containing temporal expressions	*n/a*
tmp_type	A	*location, duration, frequency, enactment, temporal_manner*	the type of the expression (defined for clauses containing temporal expressions, and assigned using a set of manually designed templates): all values except *temporal manner* are assigned according to the classification of temporal expressions available in the linguistic literature (Harkness 1987), for example *today* (location), *during those hours* (duration), *every day* (frequency), *never* (enactment); *temporal manner* is a separate pseudo-category defined to include expressions such as *immediately*, *instantly* etc.	*n/a*

clause_type	O	*assertive, imperative, infinitive, subjunctive*	the form of the main verb in the clause as output by the parser: *imperative* for clauses realized in the imperative mood, *subjunctive* for those realized in the subjunctive, *infinitive* for infinitival clauses (e.g., *He decided to go.*), *assertive* otherwise	*assertive*
nbr_of_sent	O	*continuous*	the index of the parent sentence in text	*-1*
sent_type	O	*exclaim, question, assert*	*exclaim* for clauses that are exclamations, *question* – for those that are questions and *assert* – for all others	*assert*

Table C.2. Features representing a clause in the coarse-grained dataset.

Name	Category	Possible values	Description	Default value
char_in_clause	C	*yes, no*	*yes* if the clause contains a mention of a character	*no*
is_subj_obj	C	*yes, no*	*yes* if the clause contains a mention of a character and its grammatical function is subject or direct object	*no*
modified_by_np	C	*yes, no*	*yes* if the mention of a character is present in the clause and it is modified by a noun phrase	*n/a*
nbr_after_first_mention	C	*continuous*	an integer that reflects the difference between the index of the current sentence and the sentence where the character is first mentioned (only defined for clauses containing mentions of characters)	*-1*
loc_in_prep	L	*yes, no*	*yes* if the clause contains a mention of a location embedded in a prepositional clause	*no*
loc_present	L	*yes, no*	*yes* if the clause contains a mention of a location	*no*
default_aspect	A	*state, activity, accomp, achieve*	default lexical aspect of the main verb in the clause; computed according to the privative model defined in (Dorr and Olsen 1997)	*n/a*
has_modal	A	*yes, no*	*yes* if the clause contains a modal verb	*no*
past_perfect	A	*yes, no*	*yes* if the clause is realized in past perfect tense	*no*
politeness_with_be	A	*yes, no*	*yes* if the clause contains one of the following expressions: *to be sorry, to be delighted, to be glad, to be sad*; the feature designed to help capture politeness expressions (e.g., *I am glad to see you)*	*no*
simple_past_present	A	*yes, no*	*yes* if the clause is realized in simple present or past tense	*no*
tmp_exp_long_duration	A	*no, long, short*	*long* if the clause contains a temporal expression denoting a long period of time, *short* if it contains an expression denoting a short period of time and *no* otherwise. This value is computed using *Tempo-*	*no*

			ralExpressionFeatures procedure outlined in **Appendix E**.	
is_assertive_clause	O	*yes, no*	*no* if the clause is not an assertion	*yes*
is_assertive_sent	O	*yes, no*	*no* if the parent sentence is not an assertion	*yes*
nbr_of_sent	O	*continuous*	the index of the parent sentence in text	*-1*

Appendix D. Templates for capturing temporal expressions.

Each temporal expression is characterized by a vector of features, which is shown in the parentheses following the expression. The vector contains three values. The first value is the magnitude of the expression. The second value corresponds to the type of the expression. The third value is the plurality. For brevity, I use the following notation to indicate the values of each feature:

type = {0: 'none', 1: 'location', 2: 'duration', 3: 'frequency', 4: 'enactment', 5: 'temporal_manner'}

self.magn = {0: 'none', 1: 'min', 2: 'hour', 3: 'day', : 'week', 5: 'month', 6: 'year', 7: 'year_plus'}

self.plur = {0: 'none', 1: 'single', 2: 'plural'}

The list contains the expressions in the following order: static expressions (e.g., now), the expressions with one unknown slot (e.g., this <time>) and the expressions with two or more unknown slots (e.g., <time> after <time>). The slot means the part of the expression that needs to be replaced. For instance the expression during this <time> can match during this year, during this day, during this hour. The last part of the list contains the expressions, which can fill these slots (e.g., year, day, minute, etc.).

When the system processes a sentence, it first attempts to find a static expression in it. If none is found, it searches for an expression with one empty slot. If no such expression is found, it proceeds to look for expressions with more than one unknown slot. The system only matches one expression per sentence (the first that it finds).

If the system finds an expression with one or more unknown slots, it searches through the list of expressions, which can fill the slots and attempts to find a match. If it finds a match, it takes the missing feature values (most often, the missing value is magnitude) from the values of the slot-filling expression.

I use the style of Python regular expressions when listing the templates. The sign '\s+' means one or more spaces and the sign '\b' denotes a word boundary.

Static expressions:

"\b ever \s+ since \b" [7, 2, 0]
"\b meanwhile \b" [0, 2, 0]
"\b as \s+ long \s+ as\b" [0, 2, 0]
"\b so \s+ long \s+ as \b" [0, 2, 0]
"\b forever \b" [7, 2, 0]
"\b ever \b" [7, 4, 0]
"\b never \b" [7, 4, 0]
"\b again \b" [2, 3, 0]
"\b weekly \b" [4, 3, 0]

"\b yearly \b" [6, 3, 0]
"\b monthly \b" [5, 3, 0]
"\b daily \b" [3, 3, 0]
"\b nightly \b" [3, 3, 0]
"\b hourly \b" [2, 3, 0]
"\b sometimes \b" [3, 3, 0]
"\b \s+ now \s+ and \s+ then \b" [0, 3, 0]
"\b occasionally \b" [3, 3, 0]
"\b whenever \b" [0, 3, 0]
"\b again \s+ and \s+ again \b" [0, 3, 0]
"\b at \s+ times \b" [3, 3, 0]
"\b a \s+ couple \s+ of \s+ times \b" [2, 3, 0]
"\b several \s+ times \b" [3, 3, 0]
"\b constantly \b" [0, 3, 0
"\b often \b" [3, 3, 0, 1]
"\b when \b" [0, 1, 0]
"\b sometimes \b" [0, 1, 0]
"\b once \s+ upon \s+ a? \s* time \b" [0, 1, 0]
"\b long \s+ ago \b" [7, 1, 0]
"\b long \s+ time \s+ ago \b" [7, 1, 0]
"\b not \s+ long \s+ after \b" [0, 1, 0]
"\b before \s+ that \b" [0, 1, 0]
"\b soon \b" [0, 1, 0]
"\b presently \b" [0, 1, 0]
"\b usually \b" [7, 1, 0]
"\b soon\b" [0, 1, 0]
"\b whenever \b" [0, 1, 0]
#"\b as \s+ always \b" [7, 1, 0]
"\b as \s+ soon \s+ as \b" [0, 1, 0]
#"\b as \s+ usual \b" [7, 1, 0]
"\b at \s+ once \b" [1, 1, 0]
"\b always \b" [7, 1, 0]
"\b used \s+ to \b" [7, 1, 0]
"\b currently \b" [0, 1, 0]
"\b afterwards \b" [0, 1, 0]
"\b now \b" [2, 1, 0]
"\b nowadays \b" [7, 1, 0]
"\b first \b" [0, 5, 0]
"\b already \b" [1, 5, 0]
"\b any longer \b" [0, 5, 0]
"\b momentarily \b" [1, 5, 0]
"\b finally \b" [0, 5, 0]
"\b instantly \b" [1, 5, 0]
"\b quickly \b" [1, 5, 0]
"\b suddenly \b" [1, 5, 0].

"\b no longer \b" [0, 5, 0]
"\b at last \b" [0, 5, 0]
"\b just \b" [1, 5, 0]
"\b yet \b" [0, 5, 0]
"\b exactly \b" [0, 5, 0]

Dynamic expressions:

"\b for \s+ <time>\b" [0, 2, 0]
"\b from \s+ <time>\b" [0, 2, 0]
"\b <time> \s+ since \b" [0, 2, 0]
"\b during \s+ <time> \b" [0, 2, 0]
"\b until \s+ <time> \b" [0, 2, 0]
"\b whole \s+ <time>\b" [0, 2, 0]
"\b within \s+ <time> \b" [0, 2, 0]
"\b since \s+ <time> \b" [0, 2, 0]
"\b till \s+ <time> \b" [0, 2, 0]
"\b many \s+ <time> \b" [0, 3, 2]
"\b every \s+ <time> \b" [0, 3, 2]
"\b each \s+ <time> \b" [0, 3, 2]
"\b <time> \s+ later \b" [0, 1, 0]
"\b <time> \s+ after \b" [0, 1, 0]
"\b <time> \s+ afterwards \b" [0, 1, 0]
"\b <time> \s+ before \b" [0, 1, 0]
"\b <time> \s+ ago \b" [0, 1, 0]
"\b towards \s+ <time> \b" [0, 1, 0]
"\b before \s+ <time> \b" [0, 1, 0]
"\b by \s+ <time> \b" [0, 1, 0]
"\b another \s+ <time>\b" [0, 1, 0]
"\b for \s+ the \s+ first \s+ <time>\b" [0, 1, 0]
"\b after \s+ <time> \b" [0, 1, 0]
"\b next \s+ <time> \b" [0, 1, 0]
"\b once \s+ during \s+ <time>\b" [0, 1, 0]
"\b once \s+ on \s+ <time> \b" [0, 1, 0]
"\b once \s+ or \s+ twice \s+ a? \s* <time> \b" [0, 1, 0]
"\b once \s+ upon \s+ <time> \b" [0, 1, 0]
"\b one \s+ <time> \b" [0, 1, 0]
"\b about \s+ time\b" [0, 1, 0]
"\b in \s+ <time> \b" [0, 1, 0]
"\b at \s+ <time> \b" [0, 1, 0]
"\b to \s+ <time> \b" [0, 2, 0]
"\b on \s+ <time> \b" [0, 1, 0]
"\b all \s+ <time> \b" [0, 2, 0]
"\b untill \s+ <time> \b" [0, 2, 0]

"\b during \s+ <time> \b" [0, 2, 0]
"\b some \s+ <time>\b" [0, 1, 0]
"\b that \s+ <time> \b" [0, 1, 0]
"\b this \s+ <time> \b" [0, 1, 0]

Dynamic expressions with more than one empty slot.

"\b from \s+ <time> \s+ till \s+ <time>\b" [0, 2, 0]
"\b <time> \s+ and \s+ <time>\b" [0, 1, 0]
"\b <time> \s+ after \s+ <time>\b" [0, 2, 0]
"\b <time> \s+ or \s+ <time>\b" [0, 1, 0]
"\b <time> \s+ before \s+ <time>\b" [0, 1, 0]
"\b by \s+ <time> \s+ and \s+ <time>\b" [0, 1, 0]

The expressions that can fills empty slots:

(These expressions are characterized by only two features: magnitude and plurality The sign '\w+' denotes one or more words).

"January" [5, 1],
"February" [5, 1],
"March" [5, 1],
"April" [5, 1],
"May" [5, 1],
"June" [5, 1],
"July" [5, 1],
"August" [5, 1],
"September" [5, 1],
"October" [5, 1],
"November" [5, 1],
"December" [5, 1],
"Monday" [3, 1], #days of week
"Tuesday" [3, 1],
"Wednesday" [3, 1],
"Thursday" [3, 1],
"Friday" [3, 1],
"Saturday" [3, 1],
"Sunday" [3, 1],
"Mondays" [3, 2],
"Tuesdays" [3, 2],
"Wednesdays" [3, 2],
"Thursdays" [3, 2],
"Fridays" [3, 2],
"Saturdays" [3, 2],
"Sundays" [3, 2],

```
”daylight” [ 2, 1],
” daytime” [ 3, 1],
” noontime” [ 3, 1],
”yesterday” [ 3, 1],
”to\-?day” [ 3, 1],
”to\-?morrow “ [ 3, 1],
”forth?night” [ 3, 1],
”noon” [ 2, 1],
”mid\-?night” [ 2, 1],
”long” [ 0, 0], # long ago, not long before
”now” [ 3, 1],
r”a? \s* while” [ 0, 1],
“1[7, 8, 9]\d\d” [ 6, 1], #a year 1700 - 1999
r”year \s+ of \s+ (?:\w+\s+)? 1[7, 8, 9]\d\d” [ 6, 1],    #year of grace 1700
r”(?: (?:\d{,2}? \s+) | (?:\w+\s+) ) o\'clock” [ 2, 1],     #five o'clock, 4 o'clock
r”(?:\w+\s+){,2}? morning” [ 3, 1],
r”(?:\w+\s+){,2}? mornings” [ 3, 2],
r”(?:\w+\s+){,2}? evening” [ 3, 1],
r”(?:\w+\s+){,2}? evenings” [ 3, 2],
r”(?:\w+\s+){,2}? minute” [ 1, 1],
r”(?:\w+\s+){,2}? minutes” [ 1, 2],
r”(?:\w+\s+){,2}? month” [ 5, 1],
r”(?:\w+\s+){,2}? months” [ 5, 2],
r”(?:\w+\s+){,2}? week” [ 4, 1],
r”(?:\w+\s+){,2}? weeks” [ 4, 2],
r”(?:\w+\s+){,2}? half (?:an\s+) | (?:of\s+an\s+) | (?:\-) hour” [ 2, 1],
r”(?:\w+\s+){,2}? hour” [ 2, 1],
r”(?:\w+\s+){,2}? hours” [ 2, 2],
r”(?:\w+\s+){,2}? time(?:s)?” [ 3, 0],
r”(?:\w+\s+){,2}? age(?:s)?” [ 0, 0
r”(?:\w+\s+){,2}? moment” [ 1, 1],
r”(?:\w+\s+){,2}? moments” [ 1, 2],
r”(?:\w+\s+){,2}? moment\'s \s+ \w+” [ 1, 1], #a moment's hesitation
r”(?:\w+\s+){,2}? second” [ 1, 1],
r”(?:\w+\s+){,2}? seconds” [ 1, 2],
r”(?:\w+\s+){,2}? instant” [ 1, 1],        # that instant
r”(?:\w+\s+){,2}? life” [ 7, 1],
r”(?:\w+\s+){,2}? year” [ 6, 1],
r”(?:\w+\s+){,2}? years” [ 6, 2],
r”(?:\w+\s+){,2}? day” [ 3, 1],
r”(?:\w+\s+){,2}? days” [ 3, 2],
r”(?:\w+\s+){,2}? to\-?night” [ 3, 1],
r”(?:\w+\s+){,2}? night” [ 3, 1],
r”(?:\w+\s+){,2}? nights” [ 3, 2],
r”(?:\w+\s+){,2}? future” [ 7, 0],
```

```
r”(?:\w+\s+){,2}? past” [ 7, 0],
r”(?:\w+\s+){,2}? dawn” [ 3, 1],  #at dawn
r”(?:\w+\s+){,2}? dusk” [ 3, 1],  #at dusk
r”(?:\w+\s+){,2}? of \s+ the \s+ dawn” [ 3, 1],   #darkness of the dawn
r”yesterday\'s \s+ (?:\w+\s+){,2} “ [ 3, 1], #yesterday's party
r”to\-?day\'s \s+ (?:\w+\s+){,2}“ [ 3, 1], #today's party
r” to\-?morrow\'s \s+ (?:\w+\s+){,2}? “ [ 3, 1],    #tomorrow's party
r”(?:\w+\s+){,2}? afternoon” [ 3, 1],    #hot lazy afternoon
r”(?:\w+\s+){,2}? afternoons” [ 3, 2],   #hot lazy afternnons
r”(?:\w+\s+){,2}? breakfast” [ 3, 1],    # long breakfast
r”(?:\w+\s+){,2}? lunch” [ 3, 1],
r”(?:\w+\s+){,2}? dinner” [ 3, 1],
r”(?:\w+\s+){,2}? supper” [ 3, 1],    #hot lazy afternoon
r”(?:\w+\s+){,2}? season” [ 6, 1], #this season
r”(?:\w+\s+){,2}? seasons” [ 6, 2], #seasons
r”(?:\w+\s+){,2}? autumn” [ 6, 1], #autumn
r”(?:\w+\s+){,2}? autumns” [ 6, 2], #autumns
r”(?:\w+\s+){,2}? fall” [ 6, 1], #fall
r”(?:\w+\s+){,2}? falls” [ 6, 2], #falls
r”(?:\w+\s+){,2}? winter” [ 6, 1], #winter
r”(?:\w+\s+){,2}? winters” [ 6, 2], #winters
r”(?:\w+\s+){,2}? spring” [ 6, 1],
r”(?:\w+\s+){,2}? springs” [ 6, 2],
r”(?:\w+\s+){,2}? summer” [ 6, 1],
r”(?:\w+\s+){,2}? summers” [ 6, 2],
r”(?:\w+\s+){,2}? holyday” [ 3, 1],
r”(?:\w+\s+){,2}? holydays” [ 4, 2],
r”(?:\w+\s+){,2}? week\-?end” [ 3, 1],
r”(?:\w+\s+){,2}? week\-?ends” [ 3, 2],            #on the weekends
r”(?:\w+\s+){,2}? interval” [ 0, 1],
r”(?:\w+\s+){,2}? intervals” [ 0, 2],
r”(?:\w+\s+){,2}? while” [ 0, 0]
```

Appendix E. Pseudo-code for manually composed rules.

Fine-grained dataset.

Input: *sentence*, a data structure that contains a list of intra-sentential clauses and list of features for each clause (see **Table C.1** in **Appendix C**).

Output: *True* or *False* value. *True* means that the clause is deemed to be summary-worthy and *False* – if it is not.

```
def ClassifySentence( sentence )
    earlyMentions = 5
    if char_in_sent is False :
        return False
    for each clause in sentence:
        if char_in_clause is False:
            continue // skip this clause
        if tense is future or clause_type is imperative:
            continue // skip this clause
        //otherwise the clause contains a character mention in past or present tense
        if char_indef is True:   //(e.g., a man)
            continue // skip this clause
        if nbr_after_first_mention  = = 0 //i.e., this is the first mention of this character:
            return True
        if nbr_after_first_mention <= earlyMentions and char_modified = True (e.g.,
                apposition) or (tense is past  and perfect is yes):
            return True
        if char_is_attr is yes and char_is_subj is no:
            continue // skip this clause
        // the character-related features of this clause display no strong evidence that it is
    salient
        //therefore, we only assume that it is salient if it is descriptive
        else
            return IsDescriptive(clause)


def IsDescriptive(clause):
    if main verb is 'have':
        //this is a have-clause, use a special WordNet-based procedure for it
        return ClassifyBasedOnDirectObject(clause):
    //discard clauses in passive
    if is_passive is True:
        return False
    if durative is True and dynamic is False and telic is False: // i.e., the main verb is stative
```

```
            if tmp_type is a temporal_manner expression:        // e.g., instantly
                return False
            elif tmp_type is location and tmp_magn <=  hour:    //e.g., at 2pm
                return False
            //e.g., I am glad
            elif politeness_with_be is True and main verb is 'be' and char_pronoun=1st:
                return False
            else:
                return True
        //otherwise the main verb is dynamic
        if progr is True or perf is True:
            return False
        // if we got here, the tense is either past or present simple
        // analyze temporal expressions
        if tmp_type is frequency:
            if tmp_magn >= day: // e.g., every day
                return True
            else:    // e.g. every minute
                return False
        if tmp_type is temporal_manner:      // e.g., instantly
            return False
        if tmp_type is enactment:
            return True
        if tmp_type is duration:
            if magnitude >= month:        // e.g., for months
                return True
            else if ( magnitude is day or week ) and tmp_plur is yes: // e.g., for weeks
                return True
            else: // e.g., for hours
                return False
        if tmp_type is location:
            if magnitude >= year:
                return True // e.g., in that epoch
            else:
                return False

        return False

def ClassifyBasedOnDirectObject(clause):
        stateCategories = ['cognition', 'state', 'time', 'artifact', 'attribute',
                'entity', 'measure', 'substance', 'relation', 'person',
                'group', 'location', 'feeling', 'pronoun', 'animal', 'none']
        if top WordNet category of direct object is in stateCategories:
            return True     //stative clause
        else:
```

return False //dynamic clause

Coarse-grained dataset.

def TemporalExpressionFeatures(clause):

```
    if tmp_type is frequency
       if tmp_magn is greater than an hour // e.g., every day
          return long
       else // e.g. every minute
          return short
    elif tmp_type is temporal_manner// e.g., quickly, immediately
       return short
    elif tmp_type is enactment // e.g., ever, never
       return long
    elif tmp_type is duration
       if tmp_magn is greater than month // e.g., for a month
          return  long
       elif tmp_magn <=  week  and tmp_plur is plural: // e.g.,  for days
          return long
       else  // e.g., since an hour ago
          return short
    elif tmp_type is location
       if tmp_magn is year_plus // e.g., this year
             return long
       elif tmp_magnin  <= 'hour' or tmp_magn is 'day' and tmp_plur is'single' // e.g., today
          return short
else
       return no
```

The following procedure is responsible for assigning weights to each clause in the coarse-grained dataset based on the values of its features. *FeatureWeights* is a dictionary of dictionaries that contains weights assigned for each feature value. I initially assigned the weights using the linguistic knowledge (as outlined in **Chapter 5** and **Section 6.3**) and common knowledge about fiction. I fine-tuned the weights using the training-portion of the corpus.

def AssignScore (clause):

```
featureWeights =  {
          char_in_clause: {yes: 1, no: -10},
          default_aspect: {state: 5, activity: 2, accomp: -1, achieve: -1, none: -1},
          has_modal: {yes: 1, no: 0},
          is_assert_clause: {yes: 0.5, no: -5},
```

```
            is_assert_sent: {yes: 0.5, no: -10},
            is_subj_obj: {none: 0, yes: 5, no: 0},
            modified_by_np: {none:0, yes: 5, no: 0},
            nbr_after_first_mention: {},
            nbr_of_sent: {},
            past_perfect: {yes: 5, no: 0},
            politeness_with_be: {yes: -5, no: 0
            simple_past_present: {yes: 1, no: -5},
            tmp_exp_long_dur: {no: 0, long: 4, short: -4},
            voice: {active: 1, passive: 0}
               }
//compute weights based on continuous attributes
if nbr_after_first_mention == 0:
        featureWeights[nbr_after_first_mention] = 20
elif nbr_after_first_mention <= total number of sentences in the story / 5:
        featureWeights[nbr_after_first_mention] = 4
else:
        featureWeights[nbr_after_first_mention] = -1

if nbr_of_sent <= total number of sentences in the story / 2:
        featureWeights[nbr_of_sent] = 4
else:
        featureWeights[nbr_after_first_mention] = -4
// compute the clause score
clauseScore = 0
for each featureName, featureValue in clause features
        clauseScore =+ featureWeights[featureName][featureValue]

return clauseScore
```