NAOAKI OKAZAKI

# RESEARCH ON INFORMATION AGGREGATION AND INTEGRATION FOR MULTI-DOCUMENT SUMMARIZATION

# RESEARCH ON INFORMATION AGGREGATION AND INTEGRATION FOR MULTI-DOCUMENT SUMMARIZATION

NAOAKI OKAZAKI

# 複数文書自動要約における情報の集約と統合に関する研究

岡崎 直観


Doctor of Philosophy (Information Science and Technology)

Graduate School of Information Science and Technology
Information and Communication Engineering
The University of Tokyo

Naoaki Okazaki: *Research on Information Aggregation and Integration for Multi-Document Summarization,* Doctor of Philosophy (Information Science and Technology).

SUPERVISORS:
Prof. Mitsuru Ishizuka

LOCATION:
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, JAPAN

## ABSTRACT

This thesis addresses methodologies for aggregating information and knowledge across documents, particularly addressing three research topics that are essential to an MDS system: sentence extraction, sentence ordering, and abbreviation recognition. This thesis consists of seven chapters. The first chapter presents the background, motivation, and goal of this study. The subsequent chapter provides a review of automatic text summarization. Chapter 3 presents the task definition and evaluation methodology of the Third Text Summarization Challenge (TSC-3). Chapter 4 describes a method for sentence extraction in an MDS system. Chapter 5 addresses two approaches to text structuring for extracts from multiple documents: a novel method to refine the conventional method for arranging sentences; and a machine-learning approach to aggregate the multiple criteria for further improvement. Chapter 6 presents a methodology for building an abbreviation dictionary from a large corpus. Chapter 7 includes remarks about the future directions of this work and concludes the thesis.

Chapter 4 of this thesis presents a methodology for sentence extraction. This study is based on the assumptions that: a human reader breaks a sentence into a set of information fragments to which the sentence is referring; information fragments are mutually independent; and an information fragment has an importance score. Among various sentence representations such as bag-of-words, bi-gram, tri-gram, n-gram, and FrameNet, this study proposes the use of the dependency relations of terms in a sentence. Based on the sentence representation, the problem of sentence extraction is formalized as a combinational optimization problem that determines a set of sentences containing as many important information fragments as possible. Source documents often contain redundant information. Therefore, the algorithm reduces the importance of information fragments that have been included in the sentences chosen previously. The presented system achieved a good result using the TSC-3 evaluation corpus. The comparison among sentence representations demonstrated that the proposed representation using pair-wise dependency relations performed better than either bag-of-words or co-occurrence representations.

Chapter 5 examines a method to arrange sentences that are extracted using important sentence extraction. The most common strategy for sentence ordering is chronological ordering, which arranges sentences in the order of their publication dates. However, some sentences might lose presuppositions of their original documents. The proposed method improves chronological ordering by resolving precedent information of arranging sentences to deal with problem cases that arise in chronological ordering. This study also proposes an evaluation metric that measures sentence continuity and an amendment-based evaluation task. The proposed method achieved good results in a rating task, raising poor chronological orderings by 20% to an acceptable level. Amendment-based evaluation outperformed an evaluation that compares an ordering with an answer produced by a human. The sentence

continuity metric, when applied to the amendment-based task, showed good agreement with the rating result. Although several strategies to decide sentence ordering have been proposed in previous studies, the appropriate manner in which to combine these strategies to achieve more coherent summaries remains unsolved. This chapter also formalizes four criteria to capture the association of sentences. These criteria are integrated into a single criterion using a supervised learning approach. The experimental results showed a significant improvement over existing sentence ordering strategies.

Chapter 6 addresses abbreviation recognition for MDS. In practice, no generic rules or exact patterns have been established for dealing with abbreviation creation. Consequently, abbreviation recognition is intended to extract pairs of short forms (acronyms or abbreviations) and long forms (their expanded forms or definitions) that occur in text. Except for a few studies, the literature emphasizes the examination of parenthetical expressions and location of a textual fragment with an abbreviation definition using a letter-matching algorithm. However, the letter matching approach cannot deal with a long form whose short form is arranged in a different word order, e.g., *water activity (AW)*. For this study, we assume that a word sequence is a possible long-form if the word sequence co-occurs frequently with a specific abbreviation and not with other surrounding words. Satisfying a validation rule for being a long form, the word sequence is stored in the abbreviation dictionary. This approach detects the starting point of the long form without using letter matching. This study uses a refined letter-matching algorithm that can recognize shuffled abbreviations to validate a short/long-form pair in English. The proposed method outperformed the base-line systems, achieving 99% precision and 82–95% recall on a MEDLINE evaluation corpus (biomedical English text). Even though the statistical approach is independent from the target language, the creation process of Japanese abbreviations is much more complicated than that of English ones. The strong co-occurrence does not imply that the long form is paraphrasable to its short form and vice versa. Consequently, this study also proposes a method to classify parenthetical expressions into paraphrasable and non-paraphrasable groups. The proposed method achieved a 60.6% *f*-measure on an evaluation corpus constructed from Japanese articles published by the Mainichi Newspapers and the Yomiuri Shimbun in 1998 and 1999.

# PUBLICATIONS

## JOURNAL PAPERS

1. Naoaki Okazaki and Sophia Ananiadou. Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22(24):3089–3095, December 2006.

2. Naoaki Okazaki, Santi Saeyor, Hiroshi Dohi, and Mitsuru Ishizuka. An extension of the multimodal presentation markup language (MPML) to a three-dimensinal VRML space. *Systems and Computers in Japan*, 36(14):69-80, October 2005.

3. Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. Improving chronological ordering of sentences extracted from multiple newspaper articles. *ACM Transactions on Asian Language Information Processing (TALIP) (Special Issue on NTCIR-4: Inforamtion Access towards Asian Languages)*, 4(3):321-339, September 2005.

4. Sohei Aya, Yutaka Matsuo, Naoaki Okazaki, Koiti Hashida, and Mitsuru Ishizuka. Summarization of multiple documents with rhetorical annotation. *Transactions of the Japanese Sciety for Artificial Intelligence (in Japanese)*, 20(3):149-158, May 2005.

5. Helmut Prendinger, Junichiro Mori, Santi Saeyor, Kyoshi Mori, Naoaki Okazaki, Yustinus Juli, Sonja Mayer, Hiroshi Dohi, and Mitsuru Ishizuka. Scripting and evaluating affective interactions with embodied conversational agents. *KI Zeitschrift (German Journal of Artificial Intelligence)*, 1:4-10, February 2004.

6. Naoaki Okazaki, Yutaka Matsuo, Naohiro Matsumura, and Mitsuru Ishizuka. Sentence extraction by spreading activation with similarity measure. *IEICE Transactions on Information and Systems (Special Issue on Text Processing for Information Access)*, E86-D(9):915-926, September 2003.

7. Naoaki Okazaki, Santi Saeyor, Hiroshi Dohi, and Mitsuru Ishizuka. An extension of multimodal presentation markup language MPML to a 3D VRML space. *Transactions of Institute of Electronics, Information and Communication Engineers (IEICE) (in Japanese)*, J85-D(9):1687-1694, September 2002.

## INTERNATIONAL CONFERENCES

1. Yutaka Matsuo, Naoaki Okazaki, Kiyoshi Izumi, Yoshiyuki Nakamura, Takuichi Nishimura, Koiti Hasida, and Hideyuki Nakashima. Inferring long-term user property based on users' location history. In *Proceeding 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, to appear in January 2007.

2. Naoaki Okazaki and Sophia Ananiadou. A term recognition approach to acronym recognition. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 643-650, Sydney, Australia, July 2006. Association for Computational Linguistics.

3. Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. A bottom-up approach to sentence ordering for multi-document summarization. In Proceedings of *the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 385-392, Sydney, Australia, July 2006. Association for Computational Linguistics.

4. Naoaki Okazaki and Sophia Ananiadou. Clustering acronyms in biomedical text for disambiguation. In *Proceedings of fifth international conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, May 2006.

5. Goran Nenadic, Naoki Okazaki, and Sophia Ananiadou. Towards a terminological resource for biomedical text mining. In *Proceedings of fifth international conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, May 2006.

6. Yukio Ohsawa, Naohiro Matsumura, and Naoaki Okazaki. Understanding scenarios of individual patients of hepatitis in double helical process involving KeyGraph and DSV. In *Proceedings of the Fourth IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST05)*, pages 456-469, Muroran, Japan, May 2005, Springer.

7. Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. A machine learning approach to sentence ordering for multidocument summarization and its evaluation. In *Proceedings of 2nd Internaional Joint Conference on Natural Language Processing (IJCNLP)*, Lecture Notes in Artificial Intelligence, LNAI 3651, pages 624-635, Jeju Island, Korea, October 2005. Springer.

8. Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. Improving chronological sentence ordering by precedence relation. In *Proceedings 20th Internaional Conference on Computational Linguistics (COLING 04)*, pages 750-756, Geneva, Swiss, August 2004.

9. Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. Coherent arrangement of sentences extracted from multiple newspaper articles. In *PRICAI 2004: Trends in Artificial Intelligence: 8th Pacific Rim International Conference on Artificial Intelligence, Lecture Notes in Computer Science, LNCS 3157 / 2004*, pages 882-891, Auckland, New Zealand, August 2004. Springer.

10. Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. Coherent arrangement of sentences extracted from multiple newspaper articles. In *PRICAI 2004 Pre-Conference Workshop Notes (Workshop of Language Sense on Computer)*, Auckland, New Zealand, August 2004.

11. Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. TISS: An integrated summarization system for TSC-3. In *Working Notes of the Fourth NTCIR Workshop Meeting (NTCIR-4)*, Tokyo, Japan, June 2004.

12. Gakuto Kurata, Naoaki Okazaki, and Mitsuru Ishizuka. GDQA: Graph driven question answering system - NTCIR-4 QAC2 experiments -. In *Working Notes of the Fourth NTCIR Workshop Meeting (NTCIR-4)*, Tokyo, Japan, June 2004.

13. Yukio Ohsawa, Naoaki Okazaki, Naohiro Matsumura, Aiko Saiura, and Hajime Fujie. A scenario development on hepatitis b and c. In *Second International Workshop on Active Mining (AM-2003)*, Maebashi, Japan, October 2003.

14. Naoaki Okazaki and Yukio Ohsawa. Polaris: An integrated data miner for chance discovery. In *Workshop of Chance Discovery and Its Management (in conjunction with International Human Conputer Interaction Conference (HCI2003))*, pages 27-30, Crete, Greece, June 2003.

15. Naoaki Okazaki, Yutaka Matsuo, Naohiro Matsumura, and Mitsuru Ishizuka. Sentence extraction by spreading activation with refined similarity measure. In *the 16th International FLAIRS Conference*, St Augustine, USA, May 2003.

16. Naoaki Okazaki, Sohei Aya, Santi Saeyor, and Mitsuru Ishizuka. A multimodal presentation markup language MPML-VR for a 3D virtual space. In *Workshop Proc. (CD-ROM) on Virtual Conversational Characters: Applications, Methods, and Research Challenges (in conjunction with HF2002 and OZCHI2002)*, Melbourne, Australia, November 2002.

17. Naoaki Okazaki, Yutaka Matsuo, Naohiro Matsumura, Hironori Tomobe, and Mitsuru Ishizuka. Two different methods at NTCIR3-TSC2: Coverage oriented and focus oriented. In Keizo Oyama, Emi Ishida, and Noriko Kando, editors, *NTCIR Workshop 3: Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, Tokyo, Japan, October 2002. National Institute of Informatics (NII).

18. Naoaki Okazaki, Yutaka Matsuo, Naohiro Matsumura, Hironori Tomobe, and Mitsuru Ishizuka. Extracting characteristic sentences from related documents. In *6th International Conference on Knowledge-based Intelligent Information Engineering Systems and Applied Technologies (KES2002)*, pages 1257-1261, Crema, Italy, September 2002. IOS Press/Ohmsha.

BOOK CHAPTER

1. Naoaki Okazaki and Yukio Ohsawa. Polaris: an integrated data miner for chance discovery. In Akinori Abe and Yukio Ohsawa (eds), *Readings in Chance Discovery*, Advanced Knowledge International, 2005.

2. Yukio Ohsawa, Naoaki Okazaki, Naohiro Matsumura, Akio Saiura, and Hajime Fujie. Mining scenarios for hepatitis B and C. In Ray Paton and Laura A. McNamara (eds), *Multidisciplinary Approaches to Theory in Medicine (Studies in Multidisciplinarity)*, pp. 209–231, Elsevier Science, January 2006.

*We have seen that computer programming is an art,*
*because it applies accumulated knowledge to the world,*
*because it requires skill and ingenuity, and especially*
*because it produces objects of beauty.*

— Donald E. Knuth [32]

## ACKNOWLEDGMENTS

This thesis would not have been possible without the support of many people. I would like to thank them for your continued advice and guidance.

First of all, I would like to express my sincere thanks and appreciations to my adviser, Prof. Mitsuru Ishizuka, Department of Creative Informatics, the University of Tokyo. He have been guiding my work, providing the excellent facilities, and ensuring various support throughout my studies since I was an undergraduate student.

I wish to express my warm thanks to Prof. Yukio Ohsawa, Department of Quantum Engineering and Systems Science, School of Engineering, the University of Tokyo. He introduced me to the field of Chance Discovery, which has remarkable ideals and concepts for bridging activities in a real world and information mining. I would like to acknowledge his colleagues, especially, Prof. Hiroko Shoji, Dr. Eiji Murakami, Mr. Yuki Usui, and Ms. Miwa Takayama.

Many thanks go to Prof. Jun'ichi Tsujii and Dr. Sophia Ananiadou for giving the great opportunity of my research activity at the University of Manchester, UK. I spent a quality time discussing with researchers in the National Centre for Text Mining, developing text-mining applications, extending my research topics to terminological management. I also acknowledge Prof. Hideki Mima of the University of Tokyo, Mr. John McNaught, Dr. Yoshimasa Tsuruoka, and Phillip Cotter of the National Centre for Text Mining.

I thank Dr. Yutaka Matsuo for accommodating the discussion group *Matsuo-gumi* in our laboratory specialized in studies on World Wide Web and Natural Language Processing. The discussion group has supported me with technical discussion, counseling, recreational activities, etc. I wish to acknowledge the following researchers and colleagues who have provided useful comments and suggestions on this work: Dr. Naohiro Matsumura, Jun'ichiro Mori, Gakuto Kurata, Sohei Aya, Yohei Asada, Shigeru Fujimura, YingZi Jin, Danushka Bollegala, and Kenji Hirohata of the *Matsuo-gumi*.

This study uses electronic articles published by the Mainichi Newspapers and the Yomiuri Shimbun. I wish to thank the organizers of Text Summarization Challenge (TSC) for providing good evaluation corpora for text summarization.

And finally, special thanks to my wife Satomi, parents, brothers, and friends who endured this long process with me, always offering support and love.

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# Part I

# PRELIMINARIES

# 1 INTRODUCTION

## 1.1 BACKGROUND

Numerous computerized documents are accessible on-line. In November 2006, the Netcraft Web Server Survey[1] reported that more than 100 million web sites with distinct domain names were accessible using the Internet. The Internet has doubled in size since May 2004, when the survey hit 50 million (Fig. 1). The total number of pages on the Web is estimated to be much greater than this figure because the number of pages of a web site depends on the service it provides (e.g., information portal, weblog hosting). These facts suggest an *information explosion*, a tremendous increase in the number of published documents.

*information explosion*

Meanwhile, search engines on the Web achieve moderate success in keeping up with the growth of computerized documents. Major search engines (e.g., *Google* and *Yahoo!*) claim to have indexed more than several billion pages on the Web. The verb 'google' has been added to the Merriam Webster Collegiate Dictionary, meaning "to use the Google search engine to obtain information about (as a person) on the World Wide Web". Using a search engine on the Web has become a pervasive means to obtain information at a reasonable cost and time.

Notwithstanding, users are often disappointed with the quantity of retrieved documents despite having narrowed the range of documents of interest. For instance, as many as 5,860,000 documents were retrieved by Google with a query "hijacking" (as of November 2006). That figure is understandable because numerous events related to hijacking have occurred throughout the world. We can improve the query to "hijacking All Nippon Airways 61" to narrow the search to a specific event of hijacking. Nevertheless, 10,300 documents were retrieved by Google; with 144 documents found, with the same query, in the Mainichi Newspaper articles published in 1999. A situation in which a person has too much information to use is called *information overload*, and has been regarded as a major problem of information access.

*information overload*

The information explosion has also engendered a big shift in structuring information. The more information retrieval systems play an important role in information society, the more often we must deal with documents gathered dynamically without inter-document structures, i.e. search results. Different from human-made document collections such as books and journals, we must sort out the structure of retrieved documents, e.g., what is common in the document set, what distinguishes one part of a document from another, what is the optimal order to read the documents, etc. In addition, information is increasingly being published in unstructured styles. We cannot expect a coherent story in a series of blog entries because a blogger might ramble about extraneous subjects such as products, news events, or local events. For those reasons, a mechanism to aggregate information published by

1 Netcraft Website: http://news.netcraft.com/

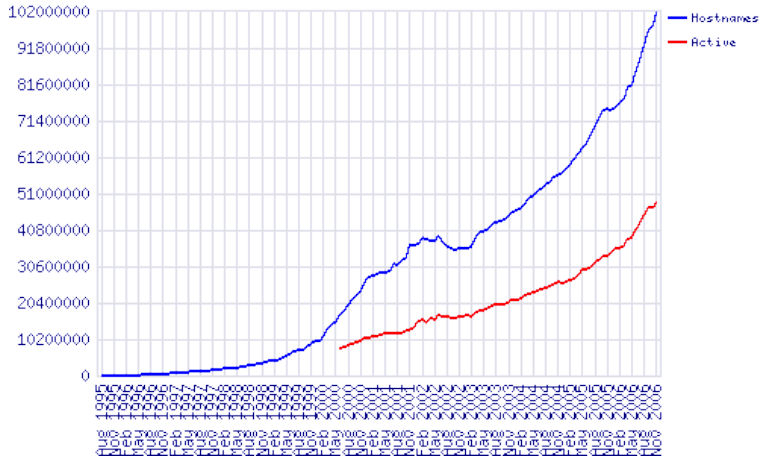



Figure 1. Total sites across all domains August 1995 – November 2006 (Netcraft Web Server Survey)

different sources is the key to a solution for the information overload problem.

*Other research topics in text mining*

To date, various research topics in text mining have made contributions to resolving the information overload problem.

TEXT CLUSTERING constructs document groups, each of which consists of documents with many words in common. The application can show pertinent topics in a document collection.

TEXT CLASSIFICATION is the task of assigning a predefined label to a given text. It is well known for its use in spam filter applications, which detect unnecessary messages in e-mail traffic.

SENTIMENT ANALYSIS detects and collects favorable/unfavorable opinions for specific subjects (e.g., organizations, products, individuals) in a text collection. Analysis results provide useful information for marketing analysis, competitive analysis, and risk management (i.e. detecting unfavorable rumors).

INFORMATION EXTRACTION is the task of filling templates (or tables) on specific items or events (e.g. terrorist events) from unstructured texts.

QUESTION ANSWERING generates, from a text collection, a direct answer to the question given in natural language.

*Automatic text summarization*

*Automatic text summarization* [41] is another challenge to the information overload problem, allowing users to control the amount of text to be read. The goal of automatic summarization is, given an information source, to present the most important content to the user in a condensed form and in a manner that is sensitive to the user's needs. We receive benefits from forms of summarization in our daily life.

ABSTRACTS of scientific articles, which are usually written by authors and are situated at the beginning of articles, are the most familiar

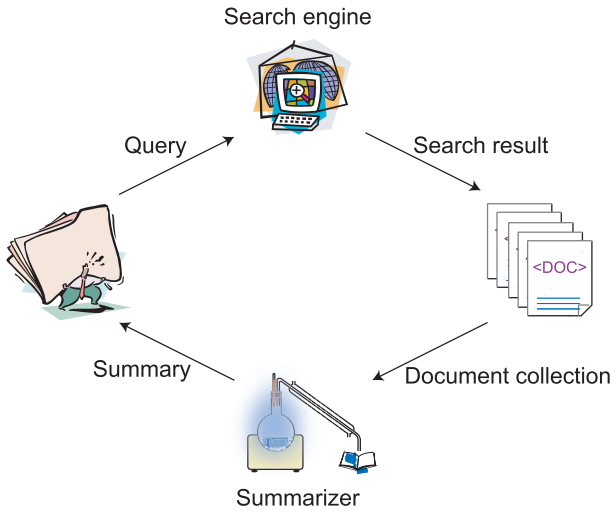



Figure 2. Multi-Document Summarization (MDS)

type of summary to researchers. Abstracts help us grasp the contents of an article in rough terms and make a decision whether further reading of the article would be fruitful.

TITLES OR HEADLINES of newspaper articles attract readers with terse expressions that suggest the body of the article. We usually choose articles in a newspaper by glancing at titles or headlines.

MINUTES of discussions or lectures are summaries to record the points of a discussion for future use.

AN RDF SITE SUMMARY (RSS) [2] is a format for exchange summaries of web contents in a standardized XML specification. The application (e.g., Mozilla FireFox, Microsoft Internet Explorer, etc.) aggregates RSS feeds from web sites and displays any updated articles.

*Multi-Document Summarization (MDS)*

Multi-Document Summarization (MDS) [66], which is a summarization task that is specialized for dealing with related documents (e.g. a collection of news stories on the same topic), has attracted much attention in recent years. The MDS research spread during the late 1990s, when information overload first came to be recognized as a serious problem that was exacerbated by the rapidly growing use of the Web. Figure 2 illustrates an approach to the information overload problem with an MDS system. In this use case, a user retrieves a set of documents that are relevant to a query expressing their information demand. In addition to the conventional process of information retrieval, the user forwards the document set to the MDS system instead of reading all of the retrieved documents. The MDS system receives all documents and presents the important contents, as a summary, to the user. The

2 RSS has different full-forms depending on its versions, RDF Site Summary (RSS 0.9 and 1.0), Rich Site Summary (RSS 0.91, RSS 1.0), or Really Simple Syndication (RSS 2.0).

user would save time and effort in reading the retrieved documents if the summary were able to satisfy user's information demand directly. The summary might also help the user determine whether they should undertake intensive reading of some of the documents.

Although the figure depicts the MDS system as one component, a practical MDS system consists of several text-processing components, e.g., document clustering, extraction, paraphrasing. Each component in an MDS system also presents research challenges in text mining. In this thesis, I specifically examine methodologies for aggregating information and knowledge across documents. More specifically, this study covers three research topics that are relevant to MDS systems.

SENTENCE EXTRACTION Sentence extraction identifies important sentences in source documents to prepare a 'material' of a summary. Aside from a few cases, sentence extraction plays a central role in MDS systems because it is impractical to produce a summary without some kind of extraction technique. Therefore, the quality of this component has a great impact on the overall MDS system performance. This study formalizes the extraction process for MDS systems as an optimization problem and shows the effectiveness of formalization.

SENTENCE ORDERING Even though sentence extraction is essential to MDS systems, it is also important to determine a coherent arrangement of sentences that have been extracted from multiple documents. A summary with improperly ordered sentences confuses a reader and degrades the quality and reliability of the summary itself. Source documents for a summary might have been written by different authors, with different writing styles, on different dates, and might also be based on different background knowledge. We cannot expect that a set of extracted sentences from such diverse documents is independently coherent. This study proposes a methodology to reconstruct the text structure for an MDS system.

ABBREVIATION RECOGNITION Abbreviations result from a highly productive type of term variation that substitutes fully expanded terms (e.g. *European Union*) with shortened term-forms (e.g. *EU*). Abbreviations hinder automatic text summarization because an abbreviation might be expressed in different forms across documents, e.g., *European Union (EU)*, *European Union* or *EU*. A good MDS system must be capable of addressing associations between shortened term forms and fully expanded terms so that a summary chooses a consistent term-form to describe the same concept or entity. In practice, no generic rules or exact patterns have been established for dealing with abbreviation creation. Consequently, an abbreviation dictionary must be used to normalize term-forms of abbreviations. This study presents a methodology for building a good quality abbreviation dictionary of abbreviations and their expanded forms by collecting definitions from numerous texts.

## 1.2 OUTLINE OF THIS THESIS

This thesis comprises six chapters (including this section). The subsequent chapter (Chapter 2) provides a review for automatic text summarization: basic notions of summarization; observation from professional summarizing; task specialization for multiple document sources; and some applications of summarization. Chapter 3 presents the task definition and evaluation methodology of the Third Text Summarization Challenge (TSC). Chapter 4 describes a methodology to sentence extraction in an MDS system. This chapter discusses a formalization of sentence extraction as an optimization problem on representations of source documents, followed by evaluation results on a test corpus. Chapter 5 addresses an approach to text structuring for extracts from multiple documents. The chapter proposes a novel method to refine the conventional method for arranging sentences and a machine-learning approach to aggregate such multiple criteria for further improvement. Chapter 6 presents a methodology for building an abbreviation dictionary from a large corpus. Chapter 7 includes remarks about future directions of this work and concludes the thesis.

# 2 AUTOMATIC TEXT SUMMARIZATION

## 2.1 INTRODUCTION

This chapter reviews automatic text summarization as a whole. Even though a typical summarization system consists of several text-processing components, this chapter does not elaborate on a specific sub-component but instead provides a comprehensive overview of text summarization. Refer to each chapter for more detailed reviews of sub-components.

This chapter is organized as follows. Section 2.2 introduces the fundamental notions that are indispensable in discussing summarization. Because human experts carry out text summarization as a professional activity, Section 2.3 explores the essence of professional summarizing to acquire useful ideas for automatic summarization. Section 2.4 dwells on Multi-Document Summarization (MDS), emphasizing the examination of additional aspects to MDS. Section 2.5 surveys existing systems that use automatic text summarization technology.

## 2.2 BASIC NOTIONS

*The goal of automatic summarization*

Mani [41] gives an excellent review of automatic text summarization. As defined in his book, the goal of automatic summarization is *to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs*. Summarization is an intellectual activity of human natural language processing. In fact, a number of expressions imply the activity of summarization: *extract*, *abstract*, *review*, *synopsis*, *outline*, *headline*, *digest*, *preview*, *minute*, *abridgment*, *compaction*, *history*, etc.

We must consider factors that specify an automatic summarization system to produce a concrete problem establishment and application. For example, Sparck-Jones [74] argues that *context factors* must be considered for development of a summarization system :

*Context factors*

INPUT What information is available as input?

OUTPUT What are the requirements for the output?

PURPOSE What is the purpose of the summaries?

In addition to these factors, various authors, e.g., Mani [41]; Hovy [29], have investigated various parameters of summarization systems.

SUMMARIZATION RATE In general, a summary is produced according to a user's length requirement. The desirable length of a summary is specifiable in several measures: the number of letters, words, sentences, or a summarization rate. For example, an academic conference sometimes requires an abstract no longer than 200 words for submitting a paper. A summarization rate (or compression rate) is defined as the ratio of summary length to the source length. Summarization ratios range

*Summarization rate*

from 0% (empty summary) to 100% (source duplication). Terms *high* and *low* are ambiguous to specify a summarization rate because a "high summarization rate" might suggest a highly condensed summary or a greater figure of the ratio. In this thesis, I express a 1% summarization rate as 'higher' than a lower rate (e.g. 99%).

*Extracts and abstracts*

FUNCTION It is important to make a clear distinction between *extracts* and *abstracts* in the context of summarization research. An extract is a summary consisting entirely of material copied from the source; an abstract is a summary containing some material that is not present in the source. In other words, when generating a summary, we allow some paraphrasing or editing from the source for abstracts, but not for extracts. A common model for an extraction process is to choose textual fragments (e.g., paragraphs, sentences, clauses, and phrases) in the source that should be included in a summary. Therefore, a computer always returns a readable and grammatical summary as long as the source text is grammatically correct. In contrast, more advanced text-processing tasks (e.g., paraphrasing, compaction, fusion) are necessary for generating high-quality abstracts.

*Indicative and informative*

RELATION TO SOURCE *Indicative* and *informative* abstracts [11] present another important distinction for summaries. An indicative abstract provides a reference function for making a decision for further reading of the original document. For example, title and sender fields of an e-mail perform a similar function to that of indicative abstracts: discard an email with a title that is likely to be an advertisement; read an email carefully if the sender is one's own supervisor. Even though indicative abstracts might present insufficient information for the original document, a user can infer the importance of the original document. In contrast, an informative abstract is expected to cover all salient information that is presented in the source. A reader should receive sufficient information of original documents from informative abstracts. For instance, an academic article usually cites a previous study along with an informative summary so that readers can continue reading a study without reading the referred article for immediately. Subtitles of TV programs and movies are also categorized as informative abstracts because the viewer can enjoy the contents only with the subtitles.

*Structured text or itemization*

COHERENCE The term 'incoherent' might evoke unwelcome outcomes from applications of natural language processing (NLP). An incoherent text does not flow because of unresolved anaphoric expressions, logical gaps, repeated information, or poor organization of the presentation. In some applications (e.g., informative abstracts, reviews), a summarization system should establish a coherence for 'polished' summaries. However, some applications might accept 'incoherent' summaries, e.g., an itemized list of words, phrases, or sentences. The coherence parameter in summarization rather refers to the method or format with which a summary is structured.

AUDIENCE Each user or application has an expectation when giving a source text to a summarization system. Some users might obtain source documents from an information retrieval system with a query, e.g., "fish

chips", or express their demand for a summary in natural language text, "How can we cook fish and chips?" *User-focused* summaries are tailored to the requirements of a particular user. Some applications might receive a query used for retrieving the input text as a representation of users' information demand and produce a *query-focused* summary . On the other hand, some summarization systems might not receive such additional representation and instead produce *generic summaries*.

*User-focused summary*

*Query-focused summary*

*Generic summary*

MEDIA A summarizer can use, as a source material, different media types such as text, audio, pictures, and movies as input, and produce a summary in these different media. This study deals only with the text medium.

*Monolingual, multilingual, and cross-lingual*

LANGUAGE A summarization system for text media can be *monolingual* (receiving a text described in one specific language and producing a summary in the same language), *multilingual* (receiving a text described in one of several languages and producing a summary in that same language), or *cross-lingual* (receiving a text described in several languages and producing a summary in a different language using information about the same event but described in different languages). This study deals only with monolingual summarization.

GENRE It is difficult to construct a summarization system that is applicable to any type of text. A summarization system should leverage the type of target text to achieve good performance and quality. For example, an extraction strategy for academic articles might differ from those for email messages. We might want to train a POS tagger with a corpus in the target domain to reduce its error. A summarization system for web pages might use a hyperlink structure on the Web and the structure of HTML elements in the target page.

*Single document or multiple document*

SPAN A summarizer might take a single document (i.e., single-document summarization) or multiple documents (i.e., multi-document summarization) as input. Multi-document summarization is described in detail later (Section 2.4).

## 2.3 PROFESSIONAL SUMMARIZING

Summarization is an important achievement of human cognition for natural language texts. Everyone performs summarization very often in everyday communications, e.g., telling funny experiences that happened yesterday. However, professional abstractors , who carry out summarization as their professional occupation have more developed techniques to produce summaries. Studying professional abstracting might provide a useful reference to automatic summarization.

*Professional abstractors*

Cremmins [16] determined four "approximately" defined stages for the abstracting process.

*Cremmins' four stages for abstracting*

EMPHASIS ON THE DOCUMENT STRUCTURE. This stage examines basic features of the target document, e.g., genre, headings, overall structure. These features support subsequent stages to locate relevant information in the document.

IDENTIFYING RELEVANT INFORMATION. In this stage, the abstractor identifies the relevant information in the source. Abstractors might search for cue phrases, e.g., "In conclusion, ..."; "The results suggest ...". Sentence position also provides a good clue: the first sentence is often a topic sentence; the last sentence often summarizes a paragraph.

EXTRACTING, ORGANIZING, AND REDUCING THE INFORMATION. Once the relevant information is identified, the abstractor extracts the important contents from the original document, reduces the contents if necessary, and arranges them in a standard order, e.g., purpose, methods, results, and conclusion.

REFINING THE INFORMATION. The last stage edits and reviews the rough abstract produced in the previous stage.

*Endres-Niggemeyer's three stages for abstracting*

Endres-Niggemeyer [19] reported an in-depth study of six expert summarizers at work, comprising a general description of their techniques and their process organization. Based on the study of empirical evidence of real-world situations, she decomposed the abstracting process into three stages as follows.

DOCUMENT EXPLORATION The first stage of the abstracting process is to collect meta-information of the target document, for example, to identify title, outline, layout, formatting, genre, and structure of the document. The knowledge obtained in this stage is used by the abstractor in later stages.

RELEVANCE ASSESSMENT In this stage, the abstractor recognizes the thematic structure of the document. The *theme*, which is *a structured representation of discourse meaning* in Endres-Niggemeyer's definition, characterizes the content of the document and helps the abstractor identify the core statements in the source document.

SUMMARY PRODUCTION This stage produces a summary mainly through cutting and pasting operations. Endres-Niggemeyer [19] explains this stage, "Their professional role tells abstractors to avoid inventing anything. They follow the author as closely as possible and reintegrate the most important points of a document in a shorter text. For this reason, we can roughly characterize their text production style as copying relevant text items from the original document, and reorganizing them to fit into a new structure, often with the help of standard sentence patterns."

Let us discuss a computational approach of summarization based on the abstracting process modeled by Endres-Niggemeyer [19]. Which kind of meta-information is available for a summarization system depends on the application. For instance, a summarization system designed for a specific news source can detect the title, genre, and structure of an article easily using simple rules. However, a summarization system for web pages might require analysis of the structure of HTML annotations to obtain the page title, or to infer the genre of the page using its content.

As for the second stage, computers also require an internal representation to interpret the source document. Although the current

NLP technology has not yet incorporated internal representation in the human mind, we might approximate it with bag-of-words, n-grams, syntactic trees, rhetorical structures, etc. Suitable representations for summarization are discussed in Chapter 4.

In the third stage, it is preferable for human abstractors to avoid any alteration of the source document. Human abstractors should not compose a sentence from scratch but rather reuse passages of the original document to the greatest degree possible. We infer from the strategy of human abstractors that a summarization system should center on extraction of relevant passages from the original document and reorganizing them to fit into a new text. Studies in this thesis contribute to these operations directly: Chapter 4 describes a formalization of extraction problem; Chapters 5 and 6 deal with text reorganization respectively at sentence and term levels.

## 2.4 MULTI-DOCUMENT SUMMARIZATION

*Multi-Document Summarization*

*Multi-Document Summarization (MDS)* is, by definition, *the specialization of document summarization to collection of related documents* (e.g., a collection of news stories about the same topic). Research on MDS has attracted much attention in recent years since information overload has come to be recognized as a serious problem, concomitant with the rapid growth of the Web. The information explosion has increased the importance of information retrieval systems. We have more occasion to deal with unstructured documents gathered dynamically using an information retrieval system. In addition, information is increasingly being published in unstructured styles: email communications, blogs, bulletin boards, etc. For that reason, a technology to view a document collection at a glance would be very useful.

*Special aspects for MDS*

In addition to research issues in single document summarization (SDS), several important aspects are noteworthy for MDS research. Summaries of three important aspects for MDS research are given in this section.

RELATION TYPE ACROSS DOCUMENTS. An MDS system should reflect how source documents are gathered. For example, some collections of newspaper articles might consist of a series of articles about an event, i.e., an article about the event and its follow-up articles. A summary for such articles would give a comprehensible overview of the event, and track the event occurrence. On the other hand, some collections of articles might consist of descriptions of the same event from several news sources with different perspectives. In this case, an MDS system is expected to include common information across documents but to exclude redundant information.

EXTRACTION FROM DIFFERENT SOURCES. An MDS system might employ the same extraction algorithm as those developed for SDS. However, the algorithm might extract the same information repeatedly because a document collection that is relevant to a query might contain overlapping information. Consequently, an MDS system is expected to extract relevant text passages but

to exclude redundant contents, detecting *common* information among the source documents and *prominent* information in each document.

AGGREGATION FROM DIFFERENT SOURCES. A single document written by one author is expected to have consistent usages of proper nouns and technical terms (aside from a few special tasks such as compilation). In contrast, multiple documents might use different expressions for referring to the same concept or entity, usually for documents written by different authors. For example, multiple documents often have terminological variants: spelling (e.g. *color* and *colour*), synonyms (e.g. *avian flu* and *bird flu*), and acronyms (e.g. *European Union* and *EU*), etc. Therefore, an MDS system should be able to process terminological variants so that a summary represents the selection of a consistent set of terms.

## 2.5 APPLICATION EXAMPLES

Automatic summarization has created numerous practical applications as products or services. For example, recent word processor products such as Microsoft Word[1], JustSystems Ichitaro[2], and Fujitsu OASYS[3] have summarization features. The IBM Intelligent Miner for Text[4] also has a summarization component integrated to the data-mining software suite.

Major search engines such as Google[5] show snippets of web pages as a part of search results. Google snippets resemble query-focused summaries of web pages because the service extracts text passages containing the query term. Snippets provide indicative summaries from which we can judge the relevance of web pages.

*Google snippets*

*Google News service*

Google News[6] is a computer-generated news site that aggregates headlines from more than 4,500 English-language news sources worldwide, groups similar stories together, and displays them according to each reader's personalized interests. The articles are selected and ranked by computers that evaluate, among other things, how often and on what sites a story appears online. Stories are sorted without regard to political viewpoint or ideology so that we can choose from a wide variety of perspectives on any given story. The service seems to employ some summarization techniques such as headline extraction, document clustering, and lead-paragraph extraction, although Google has not revealed technical details of the service.

*NewsInEssence news delivery service*

NewsInEssence (NIE)[7] [65] is a news delivery and summarization system developed at the University of Michigan. Given a user's topic specification (indicated via an example article or keywords), NIE searches across dozens of news sites to collect a cluster of related stories. It generates a summary of the entire cluster, highlighting its most important

1 Microsoft Office Website: http://office.microsoft.com/
2 JustSystems Ichitaro Website: http://www.ichitaro.com/
3 Fujitsu Co. Ltd. OASYS Website: http://software.fujitsu.com/jp/oasys/
4 IBM Intelligent Miner Website: http://www.software.ibm.com/data/iminer/
5 Google Website: http://www.google.com/
6 Google News Website: http://news.google.com/
7 NewsInEssence Website: http://www.newsinessence.com/


Figure 3. NewsInEssence

contents. In fact, NIE employs MEAD [64], which is a publicly available toolkit for multilingual summarization and evaluation. The toolkit implements multiple summarization algorithms such as position-based, TF*IDF, and query-based methods. Methods for evaluating the quality of the summaries include co-selection (precision/recall, kappa, and relative utility) and content-based measures (cosine, word overlap, bigram overlap). Figure 3 shows a screenshot of NIE as of 1 December 2006, showing clusters of news articles and their summaries.

# 3 TEXT SUMMARIZATION CHALLENGE (TSC)

## 3.1 TEXT SUMMARIZATION CHALLENGE (TSC)

Many evaluation workshops are intended for building and evaluating powerful multi-purpose information systems. For example, DARPA's Translingual Information Detection Extraction and Summarization (TIDES), ARDA's Advanced Question & Answering (AQUAINT), and NIST's Text Retrieval Conferences (TREC) cover a range of programs that specifically examine different tasks requiring their evaluation designs. In Fall 2000, TIDES sponsored a workshop to explore different ways of summarizing a common set of documents. That effort fostered continuing evaluation in the area of text summarization, as typified by the Document Understanding Conferences (DUC), sponsored by the Advanced Research and Development Activity (ARDA) and run by the National Institute of Standards and Technology (NIST). In Japan, the Text Summarization Challenge (TSC) was held as a part of the NII-NACSIS Test Collection for the IR Systems (NTCIR) project from 2001 to 2004. This section describes the task definition and evaluation methodology of TSC-3 [25] (held in NTCIR-4) because the study makes use of its evaluation corpus.

*Document Understanding Conferences (DUC)*

*Text Summarization Challenge (TSC)*

For TSC-3, 30 sets of documents were prepared. Each document set is relevant to a specific query (topic) in Mainichi and Yomiuri newspaper articles published in 1998 and 1999. The numbers of documents in a set are 5–19; the average number of documents in a set is 11.7. The topics include "a bidding war for International Digital Communications (IDC) between Nippon Telegraph and Telephone (NTT) and Cable and Wireless (C&W)," "Night Landing Practice (NLP) of ship-based aircraft of the Independence," "the cloned sheep Dolly," and "release of AIBO." Two kinds of summarization tasks were designed by TSC-3: *important sentence extraction*, in which a system extracts important sentences no longer than the specified number of sentences from a given document set; *abstracting*, in which a system produces a summary that is no longer than a specified number of characters.

### 3.1.1 *Evaluation metrics for extracts*

*Automatic evaluation*

For extracts, TSC-3 used automatic evaluation metrics. They assumed that the role of sentence extraction in MDS is to prepare a set of sentences that is suitable for producing an abstract. Multiple sentences in source documents might refer to identical information. Therefore, they noted the process of how an extract yields a sentence $a$ in an abstract $A$. If a sentence in an abstract $a_1$ derives from a sentence in an extract $s_1$, we denote that

$$a_1 \leftarrow s_1. \tag{3.1}$$

In addition, the sentence in the abstract $a_1$ might be produced as an integration of sentences in an abstract, e.g., $s_2$, $s_3$, and $s_4$. We denote

| ABSTRACT | EXTRACT |
|---|---|
| $a_1$ | $s_1 \vee (s_6 \wedge s_7)$ |
| $a_2$ | $(s_3 \wedge s_{13} \wedge s_{14})$ |
| $a_3$ | $(s_{27} \wedge s_{28} \wedge s_{29}) \vee (s_1 \wedge s_{21} \wedge s_{54})$ |

Table 1. An example of correct data of an extract

this situation with the $\wedge$ operator,

$$a_1 \leftarrow s_2 \wedge s_3 \wedge s_4. \tag{3.2}$$

The sentence $a_1$ might be produced as the combination of the abstracting processes in Formulas 3.1 and 3.2. We represent this situation with the $\vee$ operator, as

$$a_1 \leftarrow s_1 \vee (s_2 \wedge s_3 \wedge s_4). \tag{3.3}$$

In this way, they prepared correct data of an extract for each topic. Table 1 shows an example of correct data for a topic.

Hirao et al. [26] defined *precision* and *coverage* on the TSC-3 corpus. Precision is the ratio of the number of true-positive sentences over the total sentences extracted by a system. The following formula defines the precision:

*Precision*

$$\text{Precision} = \frac{m}{h}. \tag{3.4}$$

In formula 3.4, $h$ denotes the number of sentences in the minimum correct set $S^*$, where an abstract is produced by a minimum number of sentences in the evaluation corpus. The minimum-correct set $S^*$ is obtained by solving the constraint satisfaction problem (CSP). For the example shown in Table 1, the minimum-correct set $S^*$ will be

$$S^* = \{s_1, s_3, s_{13}, s_{14}, s_{21}, s_{54}\}. \tag{3.5}$$

Therefore, if a system outputs an extract $S = \{s_6, s_7, s_{10}, s_{13}, s_{29}, s_{35}\}$, the precision is computed as

$$\text{Precision} = \frac{4}{6} = 0.667. \tag{3.6}$$

*Coverage*

Coverage is an evaluation metric assessing the ratio of the number of true-positive sentences over sentences in the reference set. To define the coverage, they introduced the notation of $E_i$: each element $e \in E_i$ presents a minimum set of sentences that can completely produce an abstract $a_i$. In other words, if an extract contains all sentences in any element of $E_i$, they deem that the extract covers the sentence in the corresponding abstract $a_i$. Table 2 represents an example of a Table 1 with the notation of minimum sets.

Formulas 3.7 and 3.8 define the coverage of an extract,

$$\text{Coverage} = \frac{1}{|A|} \sum_{i=1}^{|A|} \text{cover}(a_i), \tag{3.7}$$

| ABSTRACT | EXTRACT |
|---|---|
| $E_1 = \{e_{11}, e_{12}\}$ | $e_{11} = \{s_1\}, e_{12} = \{s_6, s_7\}$ |
| $E_2 = \{e_{21}\}$ | $e_{21} = \{s_3, s_{13}, s_{14}\}$ |
| $E_3 = \{e_{31}, e_{32}\}$ | $e_{31} = \{(s_{27}, s_{28}, s_{29}\}, e_{32} = \{s_1, s_{21}, s_{54}\}$ |

Table 2. An example of correct data divided into minimum sets

$$\text{cover}(a_i) = \max_{e \in E_i} \left\{ \frac{\text{match}(e, S)}{|e|} \right\}. \tag{3.8}$$

In Formulas 3.7 and 3.8, $A$ is a set of sentences in an abstract, $|A|$ presents the number of sentences in the abstract $A$, $a_i \in A$ is an element of an abstract, $E_i$ presents a set of minimum sentence sets for an abstract $a_i$, $e \in E_i$ is an element of $E_i$ representing a minimum sentence set, $|e|$ denotes the number of sentences in the minimum set $e$, and function $\text{match}(e, S)$ presents the number of sentences in an extract $S$ that also exist in the minimum set $e$. Coverage of an extract, $S = \{s_6, s_7, s_{10}, s_{13}, s_{29}, s_{35}\}$, against the abstract of Table 2 is calculated as follows.

$$\begin{aligned} \text{Coverage} &= \frac{1}{3} \sum_{i=1}^{3} \text{cover}(a_i) \\ &= \frac{1}{3} \{\text{cover}(a_1) + \text{cover}(a_2) + \text{cover}(a_3)\} \\ &= \frac{1}{3} \left\{ \max(0, 1) + \max(\frac{1}{3}) + \max(\frac{1}{3}, 0) \right\} \\ &= 0.533. \end{aligned} \tag{3.9}$$

### 3.1.2 *Evaluation metrics for abstracts*

For abstracts, TSC-3 designed three evaluation tasks: *content coverage*; *quality questions*; and *pseudo question answering*. In content coverage evaluation, human judges match a reference abstract with a system abstract at the sentence level, and rate each sentence in the reference abstract based on the degree of agreement between sentences in the system abstract and reference abstract. The evaluation values are between 0.0 (no coverage) to 1.0 (full coverage) inclusively. Quality questions evaluate an abstract in terms of readability by asking a set of concrete questions (Table 3) to the judges. Pseudo question answering is an extrinsic evaluation metric to test whether an abstract provides information that is sufficient to answer given questions.

*Content coverage*

*Quality questions*

*Pseudo question answering*

| QID | QUESTION |
|---|---|
| q0 | How many sentences are redundant or unnecessary? |
| q1 | How many places are there where (zero) pronouns or referring expressions are used? |
| q2 | How many pronouns have missing antecedents? |
| q3 | How many proper nouns appear in an unsuitable position? |
| q4 | How many expressions have identical meanings but use different terms? |
| q5 | How many sentences are missing important constituents? |
| q6 | How many places are there where conjunctions should be supplied or conjunctions should be deleted? |
| q7 | How many unnecessary words (adverbs, adjectives, etc.) are there? |
| q8 | Does the summary have incorrect chronological ordering? |
| q9 | How many sentences require unification of the writing style (polite style or ordinary style)? |
| q10 | How many redundant verbs are there? |
| q11 | How many sentences have wrong concordant expressions are there? |
| q12 | How many sentences have incorrect word order? |
| q13 | How many incorrect inflection words are there? |
| q14 | How many complex sentences are there that had better be divided? |
| q15 | How many sentences are there that had better be unified? |

Table 3. Quality questions for readability evaluation

## 3.2 OUR SUMMARIZATION SYSTEM

Our team constructed an MDS system for Japanese newspaper articles and participated in the TSC-3 workshop. This section presents a description of the overview of the MDS system, which implemented the outcomes of this study. In addition to the three main components (i.e., sentence extraction, sentence ordering, and abbreviation extraction), other components in the system are explained briefly.

Figure 4 illustrates the architecture of the MDS system. In an off-line process, an abbreviation recognition method (presented in Chapter 6) extracts definitions of abbreviations in the whole database. This builds an abbreviation dictionary that stores the associations between abbreviations (e.g. *EU*) and their full forms (e.g. *European Union*).

This study assumes that source documents are given to the system by, for example, a document retrieval system. In the first step, CaboCha [33], an integrated tool for Japanese morphological analyzer, shallow parser, and named-entity extractor, preprocesses source documents. This step yields dependency structures of source sentences, which are to be sent to other components.

*Abbreviation normalizer*

Through reference to the abbreviation dictionary, the abbreviation normalizer replaces occurrences of abbreviations in the source documents with their normal forms, "*a full form* (*its abbreviation*)". The system leaves the first occurrence of an abbreviation as its normal form and uses the abbreviation after the first occurrence when generating an abstract. The system recognized 0.45 kinds of abbreviations and replaced 1.2 full forms with their abbreviation per a summary.

*Anaphora resolution of 'dou'*

A newspaper article often substitutes a named entity with an anaphoric expression when the named entity occurs more than twice in the article. Figure 5 presents a typical anaphoric reference with a Japanese term

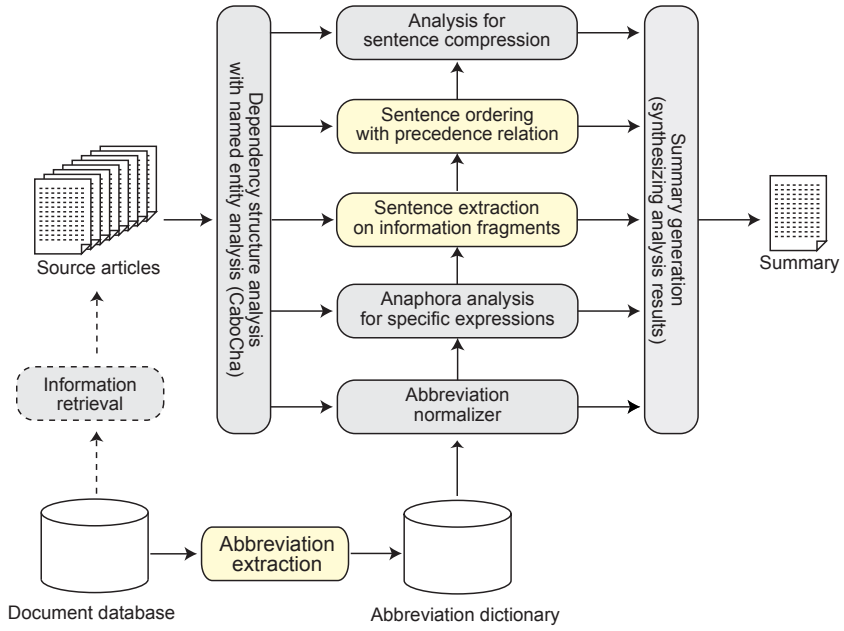



Figure 4. Architecture of the summarization system

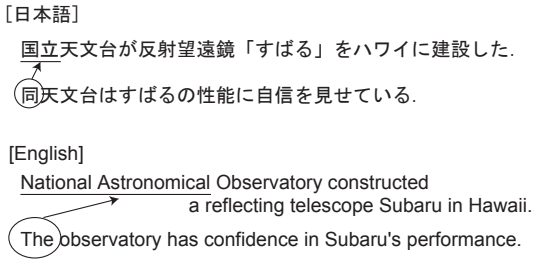



Figure 5. A typical anaphoric reference using the Japanese term 'dou'

*dou* [1]. A term *dou* is a common expression of anaphoric reference used in Japanese newspaper articles. Therefore, the system replaces it with the named entity to which the *dou* refers. The analysis component uses two constraints to find a referred named entity: the identity of the succeeding term (i.e., finding a noun phrase immediately before the term 'observatory' in the example); and the type of named entity (i.e., finding a named entity tagged as a country name when resolving an expression *the country*). The system replaced 90% of the anaphoric terms *dou* in our summary.

Sentence extraction and sentence ordering components in Fig. 4 are described respectively in Chapters 4 and 5. These components produce an ordered extract for the source documents.

*Redundant clause elimination*

Figure 6 illustrates redundant clause elimination. Extracting long

1 The meaning of *dou* is close to *the* in English, although the uses of *dou* and *the* differ greatly.

［日本語］
ソニーは６月１日よりペットロボットＡＩＢＯの予約を
インターネット上で受け付ける.
~~ソニーが１日午前９時から予約を始めた~~ペット型ロボット
ＡＩＢＯが，受け付け開始から２０分後に完売した.

[English]
Sony will accept reservations for AIBO the Entertainment Robot on the Internet on June 1st.

AIBO the Entertainment Robot for ~~which Sony started to accept reservations at 9 a.m. on the 1st~~ was sold out within 20 minutes.

Figure 6. Redundant clauses

(longer than 25 letters) clauses modifying a noun phrase, the component performs DP matching for all extracted clauses. It regards a pair of clauses that are closer than a given distance as similar clauses. In the summary generation phase, we delete clauses that are similar to previously included clauses on the basis of redundancy analysis: the system removed 3.4% letters from extracted sentences.

# Part II

# EXTRACTION

# 4 SENTENCE EXTRACTION

## 4.1 INTRODUCTION

*Passage extraction* is the most basic technology for building an automatic summarization system. With few exceptions (e.g., Banko et al. [4], Berger and Mittal [9], Daumé and Marcu [17]), summarization systems employ some kind of extractive technique. Passage extraction finds important passages in source documents to produce a summary. In general, the extraction problem is formalized as a computer-friendly task of assigning relevance scores to textual passages in source documents. Moreover, as described in the previous chapter, the production process of professional summaries can be characterized as copy-and-paste operations. Therefore, passage extraction is the low-cost but main solution to summarization research.

*Passage extraction*

*Relevance scores*

Passage extraction can be performed at any level of textual element: paragraph, sentence, clause, phrase, etc. Of those, the sentence is the most practical unit for summarization because: a sentence conveys a minimum semantic relation presented by the author; a paragraph will often contain unnecessary sentences that are not suitable for a summary; and a clause or phrase might be too fragmentary to convey an idea. Consequently, this study particularly addresses passage extraction in sentence level, i.e. *sentence extraction* .

*Sentence extraction*

Figure 7 illustrates the typical structure of a sentence extraction method for MDS. Given a set of source documents and a desirable length for the summary, the extraction method marks sentences that are considered important to readers. As the process of professional abstracting modeled by Endres-Niggemeyer [19], the extraction process is divisible into two stages: text analysis and extraction algorithm. Text analysis converts the source text, which is written in natural language, to its internal notation, with which computers can approximate the meaning. The extraction algorithm is expected to choose sentences with as much important information as possible. A reader wishes to gain an understanding of source documents in the specified length. Therefore, an extract should also refuse redundant information. The extraction algorithm determines that information is important for inclusion and that information is unimportant or redundant, given the internal representation.

This chapter presents a novel method for extracting important sentences from multiple documents. This chapter is organized as follows. The following section (Section 4.2) reviews the sentence extraction problem and previous studies to the problem. Section 4.3 describes various sentence representations for sentence extraction and their weighting methods. Section 4.4 presents a formalization of the sentence extraction task as an optimization problem. The performances of the representations for sentence extraction are compared and the tests of effectiveness of the formalization are performed in Section 4.5. This chapter ends with Section 4.6, which gives a summary of the outcomes.

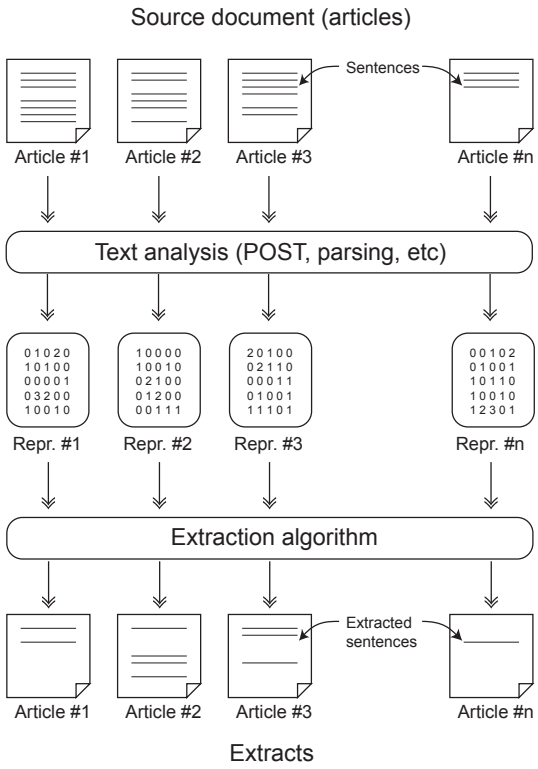



Figure 7. Typical sentence-extraction architecture

## 4.2 RELATED WORK

Humans can interpret the meaning of a text and find important spans in the text based on their interpretation. Understanding what each sentence is saying, a human discerns the types of information that are important for inclusion in a summary. However, the current NLP technology cannot deal with the meaning of a text as flexibly and efficiently as humans can. For this reason, automated extraction is fulfilled by the easier task of assessing an *importance score* to a sentence based on some surface clues in the text. In other words, an extraction method is characterized by the clue used for computing the importance score. Paice [58] examined the previous studies of automatic text summarization and classified the surface clues into seven types. Okumura and Nanba [56] rearranged that distinction with newer studies published in the 1990s. The remainder of this section summarizes the seven types of clues for sentence extraction, with some revisions.

*Clues for sentence extraction*

FREQUENCY-KEYWORD APPROACH The origin of automatic text summarization is considered to be the study undertaken by Luhn [40] in 1958. He measured the importance of words according to their frequency of use, assuming that a writer normally repeats certain words as he or she elaborates on an aspect of a subject. The proposed algorithm prepares a list of terms in a source document by removing stop

words such as pronouns, prepositions, and articles and by unifying terms that have small orthographic differences (e.g., 'similar' and 'similarity'). The algorithm counts up the frequency of occurrence of the terms and identifies important terms by removing less frequent and overly frequent terms. The score of each sentence was computed as the distinct number of important terms in the sentence.

*Term frequency*

Luhn also addressed the extensions of the algorithm: controlling summary length; weighting some terms that characterize a specific domain; application to other languages; etc. In addition, he proposed the utilization of the important terms as indexing terms for information retrieval systems: his idea also had a great influence on information retrieval. Above all, the most important extension of his statistical approach is TF*IDF [69] measure, which combines Term Frequency (TF) with Inverse Document Frequency (IDF) [31].

*Inverse document frequency*

TITLE-KEYWORD METHOD The format or meta-information of a document often indicates the important content. For example, a title (headline) of a newspaper article presents an abstract written by its author. Edmundson [18] proposed a strategy to assign a higher weight to keywords from the title (headline) than to those found only in the body of the document.

*Title heuristic*

LOCATION METHOD A document often follows the regularity of structure determined by its genre. For example, a newspaper article starts with a sentence that gives the brief description about an event, followed by detailed descriptions of the event. Some articles might conclude with a sentence describing the core argument. An academic article consists of the standard series of sections such as *background*, *aim*, *previous work*, *methodology*, *result*, and *conclusion*. An extraction method can leverage locational clues in a source document, assuming a fixed genre for source texts. Baxendale [8] noted that the first sentence in a paragraph remarks the most central points to the theme than the rest. Edmundson [18] also proposed the use of locational clues because topic sentences tend to occur very early or very late in the text of a document and in its paragraphs. With regard to summarization for newspaper articles, a *lead* strategy extracts a few sentences from the top of each article and provides the baseline method. Brandow et al. [12] reported the superiority of the lead strategy over frequency and title criteria on newspaper and magazine corpora.

*Location heuristic*

*Lead heuristic*

*Cue phrase*

CUE METHOD A *cue phrase* is one indicating the importance/unimportance of the sentence explicitly, e.g., ".. is significant", "the purpose of this study is ...", "in conclusion, ...", "for example, ...". An extraction method can examine the existence of such expressions in a sentence to estimate its importance. Edmundson [18] used a list of 783 'bonus' words (e.g., 'great', 'significant'), whose occurrence increases the sentence score, and 73 'stigma' words (e.g., 'hardly', 'impossible'), whose occurrence decreases the score. Paice [57] discussed the usefulness of text patterns (e.g., "Our investigation has shown that ...") for extraction.

TERM-RELATION APPROACH Surface clues described so far represent a sentence with a set of terms (words or phrases) in the sentence.

However, these types of surface clues cannot reflect the relations of terms revealed by a sentence (e.g., subject-verb relation, predicate-argument relation) or across sentence boundaries (e.g. lexical cohesion). The term-relation approach specifically examines various relations of terms that are stated explicitly or inexplicitly in a text. Barzilay and Elhadad [5] investigated the use of lexical chains to model the topic progression in the text. They presented an algorithm to compute lexical chains in a text using a part-of-speech tagger, a shallow parser, WordNet thesaurus. The algorithm extracts sentences with strong lexical chains as an extract. Mani and Bloedorn [42] represented a source text as a graph in which a node denotes a term and an edge presents adjacency, syntactic, synonym/hypernymy, or coreference relation. Applying a spreading activation method on the graph, the presented algorithm determines salient textual units such as words, phrases, and sentences.

*Lexical chains*

*Graph representation*

DOCUMENT-STRUCTURE APPROACH We can extend the unit of relation in the term-relation approach to broader textual spans such as clauses and sentences. Marcu [49] proposed an extraction method that captures the flow of text based on Rhetorical Structure Theory (RST). The core idea of the theory is the notion of *rhetorical relation*, which is a relation that holds between two non-overlapping text spans called *nucleus* and *satellite*, and the assumption of text coherence, which arises from a set of constraints and an overall effect that is associated with the relations. A rhetorical structure tree (RS-tree) is assembled by recursively applying individual rhetorical relations to spans that range from clause-like units to inclusion of the whole document [48]. He proposed an extraction method that assigns priority to nucleus spans over satellite and showed that nuclei in the RS-tree produce an adequate summary. Salton et al. [71] suggested paragraph extraction based on intra-document links between paragraphs. The presented algorithm yields a text relationship map from intra-document links, which indicate that the linked texts are related semantically. They proposed an extraction algorithm that starts at an important (highly bushy) node and migrates to the next most similar node at each step, with the intention of producing a readable extract. Okazaki et al. [55] also proposed sentence extraction based on similarity links between sentences. They ranked sentences by spreading activation with an assumption to produce an extract: sentences that are relevant to many other sentences of importance are also important.

*Rhetorical Structure Theory (RST)*

*Sentence similarity*

*Spreading activation*

COMBINED APPROACH An extraction method might combine multiple clues to achieve higher performance. For example, Edmundson [18] compared the performance of frequency-keyword, title-keyword, location, and cue methods and suggested an optimal combination of these clues. An optimal combination can also be determined by supervised machine-learning approaches such as Naive Bayes (e.g. Kupiec et al. [34]), Decision Tree (e.g. Mani and Bloedorn [43]), and Support Vector Machines (SVM) (e.g. Hirao et al. [24]).

## 4.3 SENTENCE REPRESENTATION

*Information fragments*

It remains a difficult challenge for NLP research to represent the meaning of a sentence. This study assumes that: a human reader breaks a sentence into a set of *information fragments* to which the sentence is referring; information fragments are mutually independent; and an information fragment has its importance score. Therefore, a sentence is approximated by a set of information fragments, each of which conveys atomic information in a sentence. I discuss sentence representation as a set of information fragments in this section and will address a formalization of sentence extraction in the next section.

*Bag-of-words*

The simplest solution for sentence representation is undoubtedly the *bag-of-words* or *vector space model* [70], which approximates the information in a sentence with a set of terms (words or phrases) contained in the sentence. For example, a number of extraction methods calculate the sentence score as the sum of term weights to estimate the importance of a sentence and the sentence similarity as the number of overlapping terms shared by the two sentences to prevent the inclusion of redundant information (e.g., Radev et al. [67]; Goldstein et al. [23]). The frequency-keyword and title-keyword approaches described in Section 4.2 also represent this type of solution.

*n-grams*

A natural extension of the bag-of-words representation would be *bi-gram*, *tri-gram*, and *n-gram*, which respectively combine two, three, and $n$ adjacent terms in a sentence to yield an unit. For example, Lin and Hovy [38] used *topic signatures*, which are uni-gram, bi-gram, and tri-gram terms clustered under related sub-topics, to score sentences.

*Co-occurrence*

Okazaki et al. [54] proposed an extraction method that represents a sentence with co-occurrences of its terms. Previous studies based on the term-relation approach (Section 4.2) also belong to this category. More "semantically rich" representations also exist, such as FrameNet [20, 3] and Script [72]. Nagao and Hasida [53] proposed a method that generates summary sentences directly from the semantic network of Global Document Annotation (GDA). This type of representation might be useful if a tool were capable of converting a source document to these representations accurately.

*Dependency relation*

Against the background of the previous work, this study proposes the use of the dependency relations of terms in a sentence. Figure 8 demonstrates a procedure for converting a sentence into a set of information fragments. A Japanese shallow parser CaboCha [33] obtains a dependency structure from a sentence. Note that the English version of a dependency tree (right side of the figure) does not reflect the syntactic structure of English source sentence because it is a word-by-word translation from the Japanese dependency tree. Deleting function words and stop words, we extract pairwise terms that have dependency relations. We obtain six pairs of terms in the Fig. 8 example.

These information fragments can be transcribed respectively into comprehensible sentences: "neutrino is an elementary particle"; "neutrino was verified"; "mass was verified"; "ICRR is a part of Japan-US Cooperative Research Group"; and "(neutrino was) verified last week". The information fragment representation of a sentence partially refers to what the original sentence is stating (with a certain degree of human interpretation). Therefore, this representation is useful to keep track of

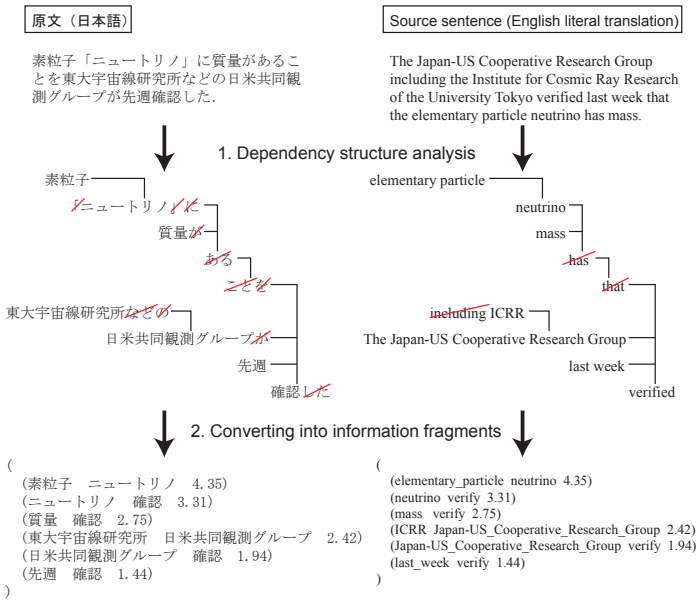



Figure 8. Generation of information fragments from a sentence

information conveyed by the extracted sentences.

Attaching a weight (importance) to each information fragment gives an indicator of which sentence contains important information and eventually that sentence we should choose for a summary. This study uses a weighting scheme based on frequency-keyword and title-keyword criteria. We introduce a function to assign higher weights for terms appearing in a headline:

$$\mathrm{hl}(x) = \begin{cases} h & (x \text{ is a headline term}) \\ 1 & (x \text{ is not a headline term}) \end{cases} \tag{4.1}$$

We set the constant $h = 2$ experimentally to double weights for headline terms, assuming that headline terms are useful as representative keywords of an article.

*Weighting information fragments*

For an information fragment that consists of terms $x$ and $y$, we prepare three kinds of weighting functions on an experimental basis:

$$\mathrm{ifw}_{\mathrm{avg}}(x,y) = \frac{\mathrm{hl}(x)\mathrm{tf}(x) + \mathrm{hl}(y)\mathrm{tf}(y)}{2}; \tag{4.2}$$

$$\mathrm{ifw}_{\mathrm{depfreq}}(x,y) = \mathrm{hl}(x)\mathrm{hl}(y)\mathrm{freq}(x,y); \text{ and} \tag{4.3}$$

$$\mathrm{ifw}_{\mathrm{combined}}(x,y) = \mathrm{ifw}_{\mathrm{avg}}(x,y) \cdot \mathrm{ifw}_{\mathrm{depfreq}}(x,y). \tag{4.4}$$

Therein, $\mathrm{tf}(x)$ denotes the occurrence frequency of term $x$ in source documents; and $\mathrm{freq}(x,y)$ represents the occurrence frequency of the

dependency relation between terms $x$ and $y$ in source documents. $\text{ifw}_{\text{avg}}(x, y)$ calculates the weight of each information fragment by average TF scores of the two terms $x$ and $y$. Also, $\text{ifw}_{\text{depfreq}}(x, y)$ estimates the weight of each information fragment by dependency frequency (i.e., the number of times in which terms $x$ and $y$ have a dependency relation in source documents). Finally, $\text{ifw}_{\text{combined}}(x, y)$ assigns combined weights of $\text{ifw}_{\text{avg}}(x, y)$ and $\text{ifw}_{\text{depfreq}}(x, y)$.

## 4.4 SENTENCE EXTRACTION FOR MDS

This section describes formalization of important sentence extraction as a combinational optimization problem that determines a set of sentences containing as many important information fragments as possible. Given a set of $n$ source sentences $C$, we consider a method to determine a subset of sentences $S^*$ that yields an extract. Formula 4.5 obtains a permutation of $k$ sentences $S^*$ that consists of a subset of the source sentences $C$, so that the score of the permutation $S^*$ is maximal of possible permutations of $k$ sentences.

*Combinational optimization*

$$S^* = \underset{s_1 \succ s_2 \succ \ldots \succ s_k}{\text{argmax}} \text{score}(s_1 \succ \ldots \succ s_k), \tag{4.5}$$

where

$$\{s_1, s_2, \ldots, s_k\} \subset C. \tag{4.6}$$

We describe the definition of the function $\text{score}(s_1 \succ \ldots \succ s_k)$ later. The number of extracted sentences $k$ is determined by either a direct constraint specified by the maximum number of sentences $K$,

*Summarization constraint*

$$k \leqslant K, \tag{4.7}$$

or a length constraint specified by the maximum length of extracted sentences $L$,

$$l = \sum_{j=1}^{k} \text{length}(s_j) \leqslant L, \tag{4.8}$$

where $\text{length}(s)$ denotes the length of sentence $s$. Formula 4.8 constrains that the sum of the lengths is no longer than $L$. Formula 4.5 applied with 4.7 extracts no more than $K$ important sentences. Formula 4.5 with 4.8 extracts important sentences no longer than $L$ words or characters (depending on the unit of the function length).

In the above discussion, we did not provide the definition of $\text{score}(s_1 \succ \ldots \succ s_k)$, which assesses the appropriateness of the permutation of sentences $s_1 \succ \ldots \succ s_k$ extracted from the source sentences $C$. Several models for measuring the appropriateness exist because this function dominates the qualification of 'important' sentences. Because formula 4.5 can be interpreted as a search problem, it is a natural assumption that the 'importance' of a sentence $s_i$ is determined by the sentences that are extracted and arranged before the sentence $s_i$. We calculate the permutation score as the sum of individual scores of sentences:

*Summary score*

$$\text{score}(s_1 \succ \ldots \succ s_k) = \sum_{i=1}^{k} \text{score}(s_i | s_1 \succ \ldots \succ s_{i-1}). \tag{4.9}$$

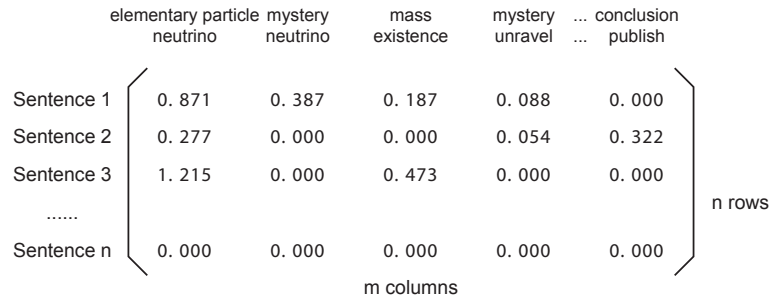



Figure 9. An example of sentence-fragment matrix

In Formula 4.9, the function $\text{score}(s_i|s_1 \succ ... \succ s_{i-1})$ presents the appropriateness of sentence $s_i$ to be extracted and arranged after sentences $s_1 \succ ... \succ s_{i-1}$.

To discuss the formalization of $\text{score}(s_i|s_1 \succ ... \succ s_{i-1})$, we introduce the $(n \times m)$ *sentence–fragment matrix* $W$ whose elements $w_{ij}$ represent the weights of an information fragments $t_j$ in sentences $s_i$. If sentence $s_i$ does not contain an information fragment $t_j$, then $w_{ij}$ is set to zero. Figure 9 illustrates an example of the sentence-fragment matrix: an information fragment consists of pairwise terms (e.g., 'elementary particle' – 'neutrino', 'mystery' – 'neutrino'); and sentence 1 has an information fragment 'elementary particle' – 'neutrino' with a weight of 0.871, but does not have the information fragment 'conclusion' – 'publish'.

*Sentence-fragment matrix*

A sentence with numerous important information fragments carries important information. Therefore, it is natural that we formalize $\text{score}(s_i|s_1 \succ ... \succ s_{i-1})$ as the weight summation of information fragments contained by sentence $s_i$, as

$$\text{score}^*(s_i|s_1 \succ ... \succ s_{i-1}) = \sum_{j=1}^{m} w_{ij}. \tag{4.10}$$

However, this extraction strategy is likely to select sentences with similar information because it does not consider redundancies of information fragments conveyed by the previous sentences $s_1, ..., s_{i-1}$. This behavior is inadequate to MDS, where source documents often contain a great deal of redundant information. Once an information fragment is presented to a reader, the importance of the fragment should decrease as the reader receives the information.

Hence, we define the function $\text{score}(s_i|s_1 \succ ... \succ s_{i-1})$ with a feature to lower weights that have already been mentioned in summary sentences:

*Sentence score*

$$\text{score}(s_i|s_1 \succ ... \succ s_{i-1}) = \sum_{j=1}^{m} \alpha^{\text{count}(t_j|s_1,...,s_{i-1})} \cdot w_{ij}. \tag{4.11}$$

In formula 4.11, the function $\text{count}(t_j|s_1 \succ ... \succ s_{i-1})$ presents the number of sentences in the preceding sentences $s_1, \succ ... \succ s_{i-1}$ that

contain the word $t_j$,

$$\text{count}(t_j|s_1 \succ ... \succ s_{i-1}) = \sum_i \text{nonzero}(w_{ij}), \tag{4.12}$$

$$\text{nonzero}(x) = \begin{cases} 1 & (x \neq 0) \\ 0 & (x = 0) \end{cases} \tag{4.13}$$

The parameter $\alpha$ ($0 \leqslant \alpha \leqslant 1$) in Formula 4.11 controls the latitude of redundant fragments in extracted sentences. We call this parameter *the duplicate information rate*. Setting $\alpha$ to 0, Formula 4.11 ignores the weights of words that are covered by the preceding sentences. Setting $\alpha$ to 1, Formula 4.11 does not discount the weights of duplicated words in extracted sentences, in other words, it is identical to the Formula 4.10. Setting $0 < \alpha < 1$, Formula 4.5 preferentially selects sentences with novel words instead of redundant ones.

*Duplicate information rate*

Let us take an example of sentence-fragment matrix (Formula 4.14) that consists of three sentences (row: $s_1$, $s_2$, $s_3$) and four information fragments (column: $t_1$, $t_2$, $t_3$, $t_4$):

$$W = \begin{bmatrix} 0.5 & 0.3 & 0 & 0.2 \\ 0.5 & 0.6 & 0 & 0.2 \\ 0 & 0.2 & 0.4 & 0.3 \end{bmatrix}. \tag{4.14}$$

Let us choose two sentences one by one from the matrix as an extract. The algorithm first extracts sentence $s_2$ (for any $\alpha$) by calculating sentence scores as

$$\begin{aligned} \text{score}(s_1) &= 0.5 + 0.3 + 0 + 0.2 = 1.0 \\ \text{score}(s_2) &= 0.5 + 0.6 + 0 + 0.2 = 1.3 \\ \text{score}(s_3) &= 0 + 0.2 + 0.4 + 0.3 = 0.9. \end{aligned} \tag{4.15}$$

Then the algorithm chooses the second sentence. Setting $\alpha = 1$ extracts sentence $s_1$, which has the second highest score, but which contains many similar information fragments to those of $s_2$. In contrast, setting $\alpha = 0.5$ discounts the importance of information fragments covered by sentence $s_2$. The algorithm will extract sentence $s_3$, which has the novel information fragment $t_3$, by computing sentence scores:

$$\begin{aligned} \text{score}(s_1|s_2) &= 0.5^1 \cdot 0.5 + 0.5^1 \cdot 0.3 + 0.5^0 \cdot 0 + 0.5^1 \cdot 0.2 = 0.5 \\ \text{score}(s_3|s_2) &= 0.5^1 \cdot 0 + 0.5^1 \cdot 0.2 + 0.5^0 \cdot 0.4 + 0.5^1 \cdot 0.3 = 0.65. \end{aligned} \tag{4.16}$$

The effects of the duplicate information rate are summarized as follows. Setting $\alpha = 0$ gives no weight to the value of information fragments covered by previous sentences; and setting $\alpha = 1$ allows the algorithm to include redundant information. Setting $0 \leqslant \alpha < 1$, the extraction algorithm favors a sentence having many novel (i.e., not included in previous sentences) information fragments because the importance of covered information fragments is lowered by Formula 4.11.

It is difficult to find an extract $S^*$, an optimal permutation of sentences $s_1 \succ s_2 \succ ... \succ s_k$, directly in Formula 4.5. Therefore, we try to find an approximate solution using a search tree: a node represents a sentence; expanding a node corresponds to a trial to choose a subsequent sentence; and summation of sentence scores from a root node to a leaf node presents the overall score of an extract.

The size of the search space would be ${}_nP_k = o(n^k)$ if we were to find an optimal extract $S^*$ that comprises $k$ sentences from $n$ source sentences. However, we can reduce neither $n$ nor $k$ in a real application: parameter $n$ is dependent on the amount of source documents, of which there might be hundreds or thousands; also, the parameter $l$ is specified directly or indirectly by a user as a summarization ratio. To find a quasi-optimal solution in a reduced search cost, we employ *beam search* method, which limits the number of branches of a node to the beam width. The implemented system controls the beam width $b (1 < b \leqslant n)$ automatically based on summary length $L$ so that the search space ($b^l$) is virtually constant. In our evaluation using the TSC-3 corpus (described later), the beam width $b$ was 3–10.

*Beam search*

## 4.5 EVALUATION

The extraction method was implemented and integrated into the summarization system for TSC-3. Refer to Section 3.1 and the TSC-3 task overview [25] for the details about the evaluation. Figure 10 reports the evaluation result of content coverage by human subjects, which assesses the quality of important sentence extraction in terms of content coverage, i.e., the extent to which a summary contains necessary information. A summary containing all necessary information will have content coverage of 1.0. The following parameter was used to set this evaluation.

- Information fragment: dependency relation of two terms
- Weighting function: average TF scores (Formula 4.2)
- Duplicate information rate: $\alpha = 0$ (deprivation of the value of covered information fragments during sentence extraction

The presented system (denoted as 'MOGS') achieved a good result on this evaluation. For both short and long summaries, the system took third place among the participating systems. The result was far better than that of a baseline system (LEAD), which features a lead extraction strategy. Our system was intended as a generic summarization system for Japanese news articles and did not employ a question-answering engine to examine questions attached to the topics specifically, e.g., "Who was engaged in the bidding war for the acquisition of IDC against NTT?" The system performed the best, 'forest', leveraged the questions with a built-in question-answering engine.

Figure 11 shows the number of redundant or unnecessary sentences per summary. The greater the degree to which a summary includes redundant information, the higher the number of redundant sentences will be. Our system (MOGS) slightly includes redundant sentences (0.067 redundant sentences for a short summary and 0.167 sentences

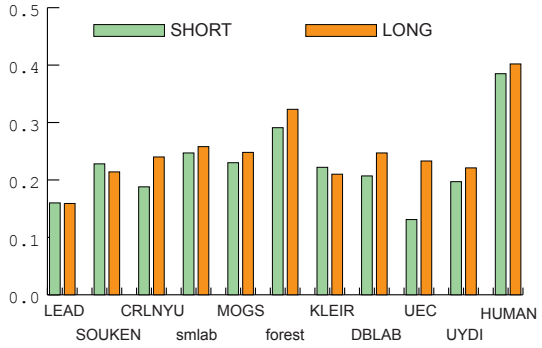



Figure 10. Results for content evaluation

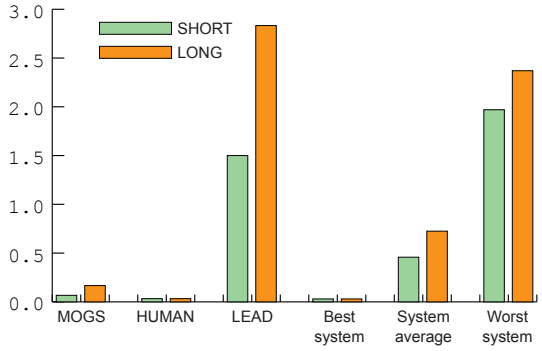



Figure 11. Number of redundant or unnecessary sentences per summary

for a long summary on average). This result shows the excellent effects of the sentence extraction.

We examined the effect of information-fragment representation by comparing six kinds of representations: *bag-of-terms* (BOT); *co-occurrence with combined score* (CO); *pairwise dependency with average term score* ($2D_{avg}$); *pairwise dependency with dependency frequency* ($2D_{depfreq}$); *pairwise dependency with combined score* ($2D_{combined}$); and and *three-pairwise dependency* (3D). Before addressing the experiment and its result, we describe these representation briefly. Bag-of-terms (BOT) represents a sentence with a set of indexing terms contained in the source sentence. For example, BOT converts the sentence in Fig. 8 into seven information fragments: 'neutrino', 'elementary particle", 'verify', 'mass', 'ICRR', 'Japan-US Cooperative Research Group', and 'last week'. We calculate the weight of an information fragment with term $x$ using the following formula:

$$\mathrm{BOTW}(x) = \mathrm{hl}(x)\mathrm{tf}(x). \tag{4.17}$$

Formulas 4.5 and 4.17 choose sentences as the common extraction methods that estimate sentence importance as a sum of term weights and sentence redundancy as vector similarity. Co-occurrence (CO) represents a sentence with a set of pairwise terms that co-occur in each

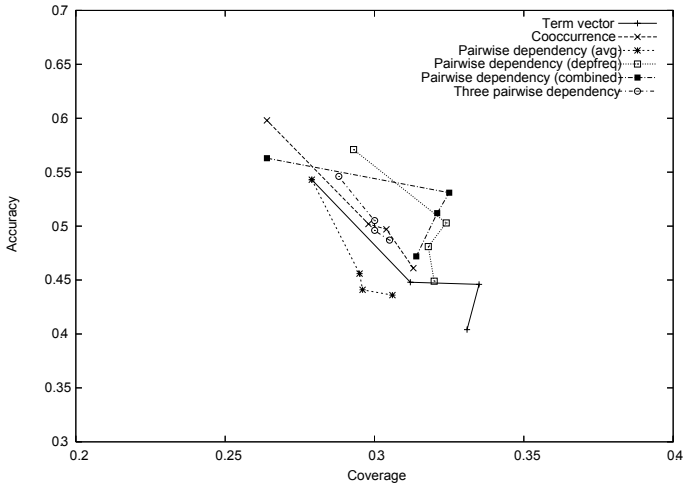



Figure 12. The effect of information-fragment representations for shorter summaries

sentence. In addition, CO converts the sentence in Fig. 8 into ${}_7C_2 = 21$ information fragments. We calculate the weight of an information fragment with terms $x$ and $y$ using the following formula:

$$\mathrm{COW}(x,y) = \frac{\mathrm{hl}(x)\mathrm{tf}(x) + \mathrm{hl}(y)\mathrm{tf}(y)}{2} \cdot \mathrm{co\_freq}(x,y). \tag{4.18}$$

In Formula 4.18, $\mathrm{co\_freq}(x,y)$ denotes the frequency of the co-occurrence relation between terms $x$ and $y$ in source documents.

Pairwise dependency (2D) is the representation described in Section 4.3. Also, $2D_{avg}$, $2D_{depfreq}$, and $2D_{combined}$ calculate the respective weights of information fragments by functions $\mathrm{ifw}_{avg}(x,y)$, $\mathrm{ifw}_{depfreq}(x,y)$, and $\mathrm{ifw}_{combined}(x,y)$. Three-pairwise dependency (3D) represents a sentence with three terms that have dependency relation. Three-pairwise dependency (3D) creates trios that consist of three terms, a term, a child of the term, and a grandchild of the term in a dependency tree. This converts the sentence in Fig. 8 example into four information fragments: 'elementary particle'–'neutrino'–'verify': 'mass'–'verify'; 'ICRR'–'Japan-US Cooperative Research Group'–'verify'; and 'last week'–'verify'. The weight of a 3D information fragment is computed using the average term score.

The author produced extracts from the TSC-3 test collection for each information-fragment representation, sequentially changing the duplicate information ratio $\alpha$ to 0.0, 0.33, 0.50, and 1.00. The extracts are evaluated in terms of precision and coverage using the metrics described in Section 3.1.1. Figure 12 shows the effect of different information-fragment representations in a high compression rate (shorter summaries). When the duplicate information ratio $\alpha$ is 1, accuracy takes the greatest value; coverage takes the smallest value. Coverage and accuracy roughly increase and decrease as the ratio $\alpha$ decreases because the extraction method rejects redundant sentences. The result showed that $2D_{depfreq}$ and $2D_{combined}$ perform better than other representations.

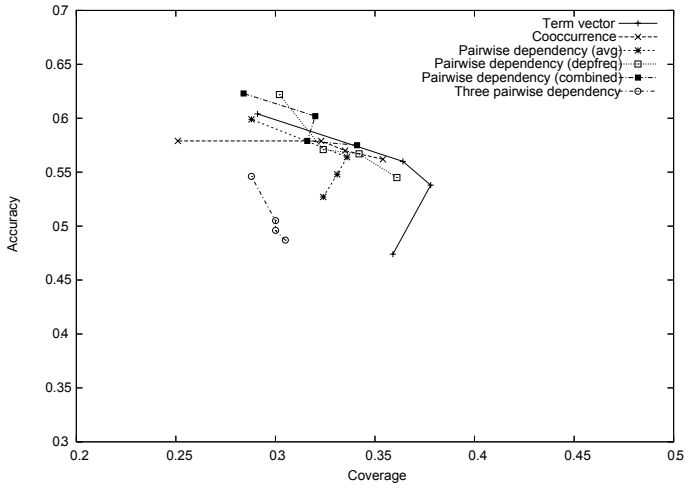



Figure 13. The effect of information-fragment representations for longer summaries

Interestingly, the bag-of-terms (BOT) performed worse than pairwise dependency (2D) but yields maximal coverage through the experiment. This result shows that representations by pairwise terms are less flexible in detection of similar information because these representations require exact matching of multiple terms. Figure 13 shows the effect of different information-fragment representations for longer summaries. All representations (except for three-pairwise dependency) perform equally well for this summarization ratio.

## 4.6 SUMMARY

This chapter described sentence representation using a set of information fragments and sentence extraction as information fragment covering. According to the TSC-3 evaluation results, the method of important sentence extraction performed well for both short and long summaries and slightly included redundant sentences. The results illustrated the excellent effects of sentence extraction. This chapter also examined the effect of information-fragment representation and described a comparison of six kinds of representation through a sentence extraction task. The experimental results suggest that the use of dependency relation improved overall quality, especially for shorter extracts.

Future research on the extraction scheme is intended to improve sentence representation. We only discounted information fragments that are included into a summary. Knowing what kinds of events tend to occur after an event, the extraction scheme can promote peripheral information fragments after inclusion of the fragments. The knowledge of natural courses of events also benefits other NLP tasks such as information extraction and sentence ordering.

# 5 STRUCTURING EXTRACTS

## 5.1 INTRODUCTION

Figure 14 illustrates an example of a typical MDS system. Given a number of documents, an MDS system yields a summary by gathering information from original documents. The summary should include as much important information as possible when a user wishes to gain an understanding of retrieved documents. Therefore, to produce an effective summary, a good MDS system must identify information in source documents to determine what sort of information is important for inclusion and which information is unimportant or redundant. As described in the preceding chapter, most existing MDS systems make use of such extraction techniques to assemble relevant textual segments in source documents. Numerous studies have examined extraction since the early stage of natural language processing (e.g. Luhn [40]) because the quality of extraction greatly affects overall performance of MDS systems.

However, post-processing of extraction is also important to secure summary readability. We should determine a proper arrangement of extracted sentences to generate a well-structured summary. We should eliminate unnecessary segments of extracted sentences to gain a higher compression ratio or insert necessary expressions to complement missing information. We should also break long sentences into several sentences or combine several sentences into one sentence. Above all, an MDS system must properly address sentence ordering. Barzilay et al. [7] conducted experiments to examine the impact of sentence ordering on summary readability. That study showed that sentence ordering markedly affects readers' text comprehension. A sentence's position in the original document, which yields a good clue to sentence arrangement for single-document summarization, is insufficient for multi-document summarization because we must simultaneously consider inter-document order. Therefore, it is necessary to establish a good ordering strategy for MDS.

*Sentence ordering*

*Sentence compression*

*Sentence splitting and fusion*

This chapter examines a method to arrange sentences that are extracted by important sentence extraction. This chapter is organized as follows. The following section (Section 5.2) reviews the sentence-ordering problem in MDS and previous studies to the problem. Section 5.3 discusses an evaluation methodology for sentence ordering and proposes new metrics, *sentence continuity* and *average continuity*, to measure the closeness between an ordering and its reference. Section 5.4 reviews the strategy of chronological ordering, which has been the baseline method in MDS systems for newspaper articles. Section 5.5 presents an approach to generate an acceptable ordering by resolving precedence relations. Section 5.6 extends the approach by integrating several criteria for sentence ordering with a machine-learning approach. This chapter is concluded in Section 5.7.

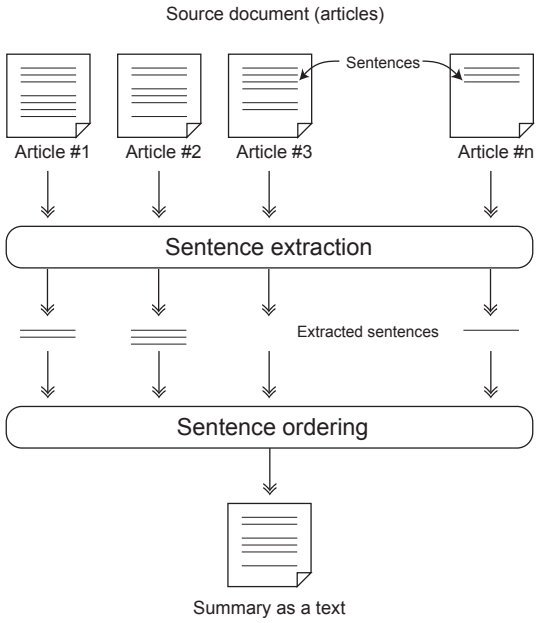



Figure 14. An extraction-based MDS system

## 5.2 SENTENCE ORDERING PROBLEM

Our goal is to determine a most probable permutation of sentences, in other words, to reconstruct a discourse structure of sentences gathered from multiple sources. Figure 15 is an example of sentence ordering problem. A possible answer to this problem would be arranging sentence $a$ at the beginning, followed by sentences $c$ and $b$ in this order. Using notation $a \succ b$ to represent that sentence $a$ precedes sentence $b$, we could express the order $a \succ c \succ b$.

When asked to arrange sentences, a human may perform this task without difficulty just as we write out thoughts in a text. However, we must consider what accomplishes this task because computers are, by their nature, unaware of ordering. Discourse coherence, typified by rhetorical relation [46] and coherence relation [28], are helpful to resolve this question. Hume [30] claimed that qualities from which association arises and by which the mind is conveyed from one idea to another are three: *resemblance*; *contiguity in time or place*; and *cause and effect*. That is to say, we should organize a text from fragmented information on the basis of topical relevancy, chronological sequence, and a cause-effect relation. The fact is especially true for sentence ordering of newspaper articles because we must typically arrange a large number of time-series events that are related to several topics.

*Discourse coherence*

*Association qualities*

*Chronological ordering*

The strategy for sentence ordering that most MDS systems use is *chronological ordering* [50, 37], which arranges sentences in the order of their publication dates. Barzilay et al. [7] addressed the problem of sentence ordering in the context of multi-document summarization. Through their experiment, they demonstrated the remarkable impact

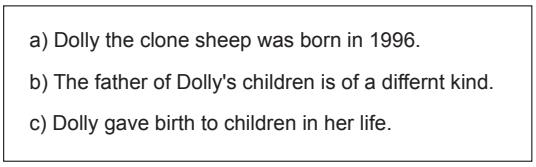



Figure 15. Arrange these sentences in the optimal order

of sentence ordering on summary readability. They proposed two naive sentence-ordering techniques such as majority ordering (examines ordering according to relative frequency in the original documents) and chronological ordering (orders sentences by publication date). Illustrating that naive ordering algorithms do not produce satisfactory orderings, Barzilay and colleagues also used human experiments to identify orderings of patterns that can improve the algorithm. Based on those experiments, they proposed another algorithm that utilizes topical segmentation and chronological ordering. They asked human judges to grade summaries produced by majority ordering, chronological ordering, and the proposed method. Results showed remarkably improved quality of orderings from the chronological ordering to the proposed method.

*Topical segmentation*

Lapata [35] proposed an approach to information ordering. She introduced three assumptions for learning constraints on sentence order from a corpus of domain specific texts: the probability of any sentence is dependent on its previously arranged sentences; the probability of any given sentence is determined only by its previous sentence; and transition probability from a sentence to its subsequent sentence is estimated by the Cartesian product defined over the features expressing the sentences. For describing sentences by their features, she used verbs (precedent relationships of verbs in the corpus), nouns (entity-based coherence by keeping track of the nouns), and dependencies (structure of sentences). Lapata also proposed the use of Kendall's rank coefficient for an automatic evaluation that quantifies the difference between orderings produced by the proposed method and a human. Although she did not describe performance comparison of the proposed method with chronological ordering, her approach is applicable to documents without publication dates.

*Probabilistic structuring*

*Kendall's rank coefficient*

Barzilay and Lee [6] investigated the utility of domain specific *content structure* for representing topics and topic shifts. They applied content models to sentence ordering and extractive summarization. Content models are Hidden Markov Models (HMMs) wherein states correspond to types of information characteristics to the domain of interest (e.g., earthquake magnitude or previous earthquake occurrences) and state transitions capture possible information-presentation orderings within the domain. They employed an EM-like Viterbi re-estimation procedure that repeats: *creating topical clusters of text spans*; and *computing models of word distributions and topic changes from the clusters*. Creating initial topical clusters by complete-link clustering via sentence similarity (cosine coefficient of word bigrams), they constructed a content model: a state represents a topical cluster; the state's sentence-emission

*Content model based on HMM*

probabilities are estimated as the product of word-bigram probabilities; and state-transition probabilities are estimated by how sentences from the same article are distributed across clusters. Barzilay and colleagues conducted an experiment of ordering sentences that were unseen in test texts and arranged in the actual text. They proposed the use of an *original-source-order (OSO) prediction rate*, which measures the percentage of test cases in which the model under consideration yields the highest probability to the OSO from among all possible permutations, along with Kendall's metric. The evaluation result showed that their method outperformed Lapata's method [35] by a wide margin. They did not address performance comparison with chronological ordering because they did not apply their approach to sentence ordering for MDS.

These previous attempts are classifiable into two groups: use of chronological information [50, 37, 7]; and learning natural ordering of sentences from large corpora [35, 6]. Advantages of the former group are that such methods are fast, easy-to-implement, and suitable for newspaper articles. Methods in the latter group are applicable to various source documents.

## 5.3 EVALUATION METHODOLOGY

This section discusses evaluation methods for sentence ordering. Evaluation methods for various NLP tasks are classifiable to human-intensive and semi-automatic approaches. Human-intensive approach, e.g., subjective grading, can measure the quality of sentence orderings accurately, but requires human efforts every time we evaluate a sample. In contrast, semi-automatic evaluation can reuse the effort involved in preparing the initial evaluation corpus. In this section, I describe human-intensive approach followed by semi-automatic approach.

### 5.3.1 *Subjective grading*

*Subjective grading*

Subjective grading is an evaluation task in which human judges mark an ordering of summary sentences. This study employs four-scale grading (Figure 16) with a clear criteria as follows. A perfect (score = 4) summary is a text that we cannot improve any further by re-ordering. An acceptable (score = 3) summary is one that makes sense and is unnecessary to revise even though there is some room for improvement in terms of readability. A poor summary (score = 2) is one that loses a thread of the story at some places and requires minor amendment to bring it up to an acceptable level. An unacceptable summary (score = 1) is one that leaves much to be improved and requires overall restructuring rather than partial revision. Judges are informed that summaries were made of a same set of extracted sentences and only sentence ordering made differences between the summaries to avoid any disturbance in rating.

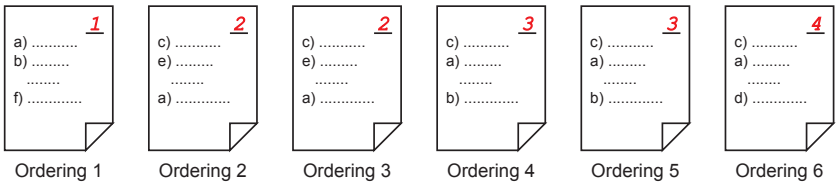



Figure 16. Subjective grading for sentence orderings

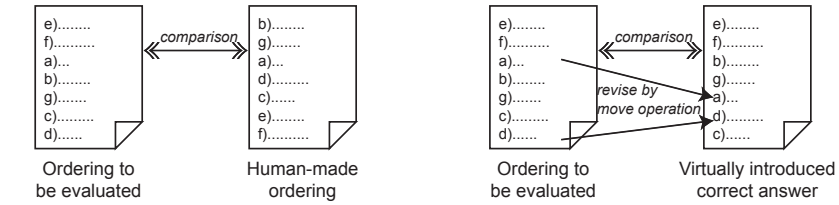



Figure 17. Automatic evaluation of sentence ordering

### 5.3.2 *Semi-automatic evaluation*

*Comparison with a reference*

In addition to the subjective grading, it is useful that we examine how close an ordering is to an optimal one. For example, we prepare a correct ordering and measure closeness of an ordering to the reference one (Figure 17-(a)). We can repeat this evaluation process once an evaluation corpus is prepared.

*Comparison with an amendment*

However, this task may be too simplistic to accept several sentence-ordering patterns for a given summary. We infer that it is valuable to measure the degree of correction required for an ordering since the task virtually requires a human corrector to mentally prepare a correct answer for each ordering. Thus, an alternative approach for semi-automatic evaluation is to ask human judges to illustrate how to improve an ordering of a summary when he or she marks the summary as *poor* in the subjective grading. Applicable operations for corrections are restricted to move operations to maintain minimum correction of the ordering. We define a move operation here as removing a sentence and inserting the sentence into an appropriate place (Figure 17-(b)).

*Comparison metrics*

The remainder of the semi-automatic evaluation entails the comparison of an ordering with its reference ordering. Figure 18 shows an ordering of nine sentences (denoted by $\{a, b, ..., i\}$) and its reference (correct) ordering. Supposing a sentence ordering to be a rank, we can convert a sentence ordering into a permutation which represents the rank of each sentence. Let $\pi$ be a permutation of an ordering to be evaluated and $\sigma$ be its reference ordering. Expressing sentences $a$ in 1, $b$ in 2, ..., $i$ in 9 respectively, we obtain permutations $\pi$ and $\sigma$ for Figure 18:

*Rank representation of an ordering*

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 6 & 7 & 2 & 3 & 4 & 5 & 8 & 9 & 1 \end{pmatrix}, \tag{5.1}$$

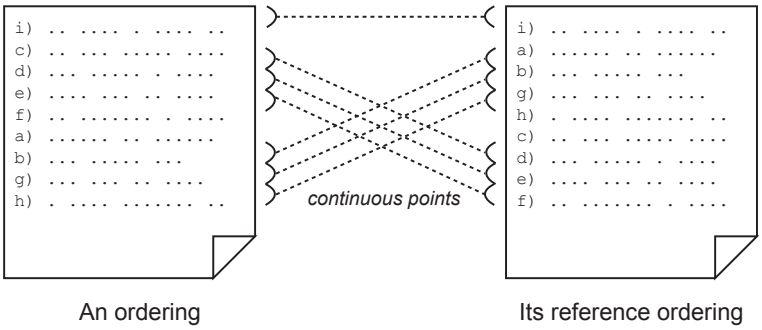



Figure 18. An ordering and its reference ordering

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 2 & 3 & 6 & 7 & 8 & 9 & 4 & 5 & 1 \end{pmatrix}. \tag{5.2}$$

*Spearman's and Kendall's rank correlations*

The above formulation transforms closeness measurement of two orderings into calculation of rank correlation of two permutations $\pi$ and $\sigma$. Spearman's rank correlation $\tau_s(\pi, \sigma)$ and Kendall's rank correlation $\tau_k(\pi, \sigma)$ are known as famous rank correlation metrics:

$$\tau_s(\pi, \sigma) = 1 - \frac{6}{n(n+1)(n-1)} \sum_{i=1}^{n} (\pi(i) - \sigma(i))^2 \tag{5.3}$$

$$\tau_k(\pi, \sigma) = \frac{1}{n(n-1)/2} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \operatorname{sgn}(\pi(j) - \pi(i)) \cdot \operatorname{sgn}(\sigma(j) - \sigma(i)). \tag{5.4}$$

Therein: $n$ represents the number of sentences; and $\operatorname{sgn}(x) = 1$ for $x > 0$ and $-1$ otherwise. These metrics range from $-1$ (an inverse rank) to 1 (an identical rank) via 0 (a non-correlated rank). For Formulas 5.1 and 5.2, we obtain $\tau_s(\pi, \sigma) = -0.07$ and $\tau_k(\pi, \sigma) = 0.11$ (i.e., the two ranks are approximately non-correlated). Spearman's rank correlation considers the absolute relation of ranking (i.e., absolute position of sentences), and Kendall's rank correlation considers the relative relation of ranking (i.e., relative position of pairs of sentences). Lapata [35] and Barzilay and Lee [6] adopted Kendall's rank correlation for their evaluations, considering that it can be interpreted as the minimum number of adjacent transpositions needed to bring an order to the reference order.

Let us examine the orderings in Figure 18 carefully. Spearman's rank correlation and Kendall's rank correlation indicate that they are non-correlated ranks. However, we notice that the reference ordering can be generated from the ordering by moving a *group* of sentences c, d, e, f to the position just after sentence h. Although a reader may find the group of sentences c, d, e, f to be incorrectly positioned, he or she

does not lose the thread of the summary because sentences within two groups, $\{c, d, e, f, \}$ and $\{a, b, g, h\}$, are arranged properly.

Sentences in a document are aligned one-dimensionally: a reader brings together continuous sentences in a text into his or her mind and interprets their meaning. In other words, when reading a text, a reader prefers local cohesion or sentence continuity as a relative relation of discontinuous sentences. Kendall's rank correlation equally penalizes inverse ranks of sentence pairs that are mutually distant in rank (e.g., sentences c and a, c and b, etc). Therefore, we propose another metric to assess the degree of *sentence continuity* in reading. We define sentence continuity as the number of continuous sentence pairs divided by the number of sentences:

*Sentence continuity*

$$\text{sentence\_continuity} = \begin{cases} (c+1)/n & \text{(if the first sentences are identical)} \\ c/n & \text{(otherwise)} \end{cases}. \tag{5.5}$$

Therein, $c$ represents the number of continuous sentence pairs. Although there is no sentence prior to the first sentences, we want to measure the appropriateness of the first sentence as a leading sentence. Hence, we define sentence continuity of the first sentence as an agreement of the first sentences between an ordering and its reference. This metric ranges from 0 (no continuity) to 1 (identical). The summary in Figure 18 may interrupt a human's reading after sentences $i$, $f$ as the human searches for the next sentence to read. We observe six continuities and an agreement of the first sentences and calculate sentence continuity: $7/9 = 0.78$.

Sentence continuity can be expressed through permutations:

$$\tau_c(\pi, \sigma) = \frac{1}{n} \sum_{i=1}^{n} \text{equals}\left(\pi\sigma^{-1}(i), \pi\sigma^{-1}(i-1)+1\right). \tag{5.6}$$

Therein, $\pi(0) = \sigma(0) = 0$; $\text{equals}(x, y) = 1$ when $x$ equals $y$ and 0 otherwise. $\sigma^{-1}(i)$ represents a sentence (or index number) of the $i$-th order in the reference; and $\pi\sigma^{-1}(i) = \pi(\sigma^{-1}(i))$ represents a rank in an ordering to be evaluated of the sentence arranged in the $i$-th order in the reference. Hence, $\text{equals}(\pi\sigma^{-1}(i), \pi\sigma^{-1}(i-1)+1) = 1$ when sentences of $(i-1)$-th and $i$-th order in the reference are also continuous in an ordering.

We can extend the sentence continuity metric to $n$-continuous sentences: the quality of a sentence ordering can be estimated by the number of continuous sentences that are also reproduced in the reference sentence ordering. This is equivalent to measuring a precision of continuous sentences in an ordering against its reference ordering. We define $P_n$ to measure the precision of $n$ continuous sentences in an ordering to be evaluated,

$$P_n = \frac{m}{N-n+1}. \tag{5.7}$$

Therein, $N$ is the number of sentences in a text; $n$ is the length of countinous sentence on which we are evaluating; $m$ is the number

of continuous sentences that appear in both evaluation and reference orderings. In Figure 18 example, the precision of 3 continuous sentences $P_3$ is calculated as:

$$P_3 = \frac{2}{5-3+1} = 0.67. \tag{5.8}$$

*Average continuity*

The Average Continuity (AC) is defined as the logarithmic average of $P_n$ over 2 to $k$:

$$AC = \exp\left(\frac{1}{k-1}\sum_{n=2}^{k}\log(P_n+\alpha)\right). \tag{5.9}$$

Therein, $k$ is a parameter to control the range of the logarithmic average; and $\alpha$ is a small value in case $P_n$ is zero. We set $k = 4$ (i.e., more than five continuous sentences are not measured for evaluation) and $\alpha = 0.01$. Average Continuity becomes 0 when evaluating and reference orderings share no countinous sentences and 1 when the two orderings are identical. BLEU metric proposed by Papineni et al. [60] for the semi-automatic evaluation of machine translations bears an analogous format to equation 5.9; in BLEU, a reference translation is compared against a machine translation using n-grams of words. In Figure 18 example, Average Continuity is calculated as 0.63.

*BLEU metric*

## 5.4 CHRONOLOGICAL ORDERING

It is difficult for computers to find a resemblance or cause-effect relation between two phenomena: numerous possible relations must be classified in detail; moreover, we do not have conclusive evidence whether a pair of sentences that we arbitrarily gather from multiple documents have some relation. A newspaper usually broadcasts descriptions of novel events that have occurred since the last publication. Hence, the publication date (time) of each article turns out to be a good estimator of the resemblance relation (i.e. we observe a trend or series of relevant events in a time period), contiguity in time, and a cause-effect relation (i.e. an event occurs as a result of previous events).

*Chronological ordering*

Figure 19 illustrates the idea of chronological ordering. Let us suppose that a sentence extraction method chose seven sentences in the source articles, $\{a, b, c, d, e, f\}$, and that we would like to determine a coherent arrangement of these sentences. The chronological ordering algorithm arranges the articles in temporal order of their publication dates. For sentences having the same publication date, the algorithm determines the order based on the sentence position, restoring original orderings in the source articles. The final ordering determined by the chronological ordering algorithm would be, $(c \succ d \succ e) \succ (f) \succ (a \succ b)$.

Lin and Hovy [37, 38] constructed an MDS system (NeATS) and arranged sentences in chronological order. Using some rules for actual date estimation, they also resolved relative temporal expressions (e.g. "Monday" or "yesterday") that point to a specific date/time in newspaper articles. If these expressions were not replaced with actual dates, the summary might mislead the reader because they might lose absolute time references during sentence extraction. Although resolving

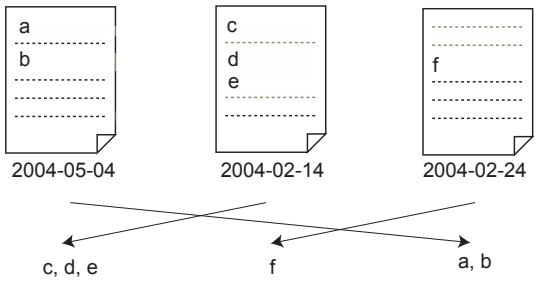



Figure 19. Chronological ordering

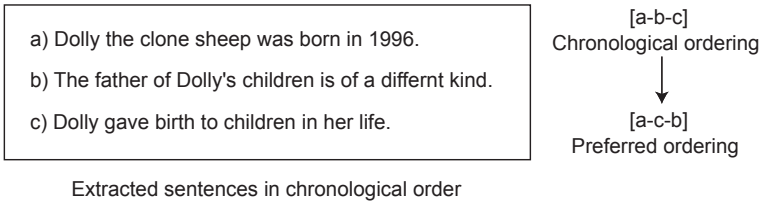



Figure 20. A problem case of chronological sentence ordering

temporal expressions in sentences (e.g. Mani and Wilson [44]; Mani et al. [45]) may allow more precise estimation of sentence relations, it still remains a difficult task.

*Problem of chronological ordering*

Let us consider the example shown in Figure 20. There are three sentences, $a, b, c$, that are extracted from different documents and refer to the clone sheep Dolly. Suppose that we infer an order $a \succ b \succ c$ by chronological ordering. When we read these sentences in this order, we find that sentence $b$ is positioned incorrectly because sentence $b$ is written on the presupposition that the reader may know that Dolly had children. An interpretation of this situation is that there were some precedent sentences prior to sentence $b$ in the original document, but sentence extraction did not choose such sentences as summary candidates. Lack of presupposition obscures what a sentence is intended to convey, thereby confusing readers. Although we may hit upon a possible solution by which we include such preceding sentences into summary candidates as an exceptional case, the solution is not appropriate in terms of stability (i.e., preceding sentences are not always required) and redundancy (i.e., including sentences may generate redundant summaries). Hence, it is not a practical solution to expand the output of the sentence extraction, which is presumed to be tuned independently.

## 5.5 LEVERAGING PRECEDENCE RELATIONS

### 5.5.1 *Precedence relation*

In order to deal with the problem case with chronological ordering, this section proposes the use of precedence relations for coherent ar-

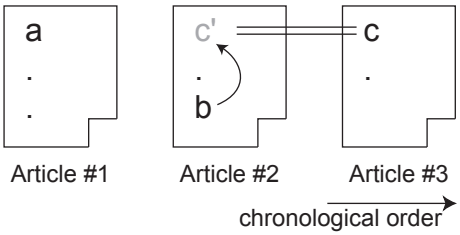



Figure 21. Background idea of ordering refinement by precedence relation

rangement of sentences. Take the chronological ordering with topical segmentation proposed by Barzilay et al. [7], this approach considers its practical refinement using in-document precedence relations. Let us observe the example of Figure 20 again. When reading sentence c, we note that it can include presuppositional information of sentence b. In addition, the sentence c requires no presupposition other than Dolly's existence, which was already mentioned in sentence a. Based on the analysis, we can refine the chronological order and revise the order to $a \succ c \succ b$, putting sentence c before sentence b. This revision enables us to assume sentence b to be an elaboration of sentence c; thereby, we improve summary readability.

Figure 21 shows the basic idea of ordering refinement using a precedence relation. Just as in the example shown in Figure 20, we have three sentences a, b, and c in chronological order. First, we select sentence a out of the sentences and check its antecedent sentences. Seeing that there are no sentences prior to sentence a in article #1, we deem it acceptable to put sentence a here. Then we select sentence b from the remaining sentences and check its antecedent sentences. This time, we find several sentences before sentence b in article #2. Grasping what the antecedent sentences are saying by means of cosine similarity of sentence vectors, we confirm first of all whether their subject content is mentioned in previously arranged sentences (i.e., sentence a). If it is mentioned, we put sentence b here and extend the ordering to $a \succ b$. Otherwise, we search for a substitution for what the precedent sentences are saying from the remaining sentences (i.e., sentence c in this example). In the Figure 21 example, we find that sentence a is not referring to what sentence $c'$ is saying, but that sentence c is approximately referring to that content. Putting sentence c before b, we finally achieve the refined ordering $a \succ c \succ b$.

As the criterion for selecting the sentence to be inserted, we introduce *distance* to put a sentence after previously arranged sentences. We define the distance as dissimilarity derived from cosine similarity between a vector of the arranging sentence and a vector of its preceding sentences. When a sentence has preceding sentences and their content is not mentioned by previously arranged sentences, this *distance* will be high. When a sentence has no precedent sentences, we define the *distance* to be 0.

Figure 22 illustrates how our algorithm refines a given chronological ordering $a \succ b \succ c \succ d \succ e \succ f$. In the Figure 22 example, we do not change the position of sentences a and b because they do not

have precedent sentences in their original article (i.e., they are lead sentences, which appear at the beginning of an article). On the other hand, sentence c has some preceding sentences in its original document. This fact presents us with a choice: we should check whether it is safe to put sentence c just after sentences a and b; or we should arrange some sentences before sentence c as a substitute for the precedent sentences. Preparing a term vector of the precedent sentences, we seek a sentence or a set of sentences that is the closest to the precedent content in sentences $\{a, b\}$, d, e, and f by the *distance* measure defined above. In other words, we assume sentence ordering to be $a \succ b \succ X \succ c$ and find appropriate sentence(s) X, if any. Supposing that sentence e in Figure 22 describes similar content as the precedent sentences for sentence c, we substitute X with $Y \succ e$. We then check whether we should put some sentences before sentence e or not. Given that sentence e is a lead sentence, we leave Y as empty (i.e., *distance* is 0) and fix the resultant ordering to $a \succ b \succ e \succ c$.

Then we consider sentence d, which is not a lead sentence, again. Preparing a term vector of the precedent sentences of sentence d, we search for a sentence or a set of sentences which is closest to the precedent content in sentences $\{a, b, e, c\}$, f. Supposing that either sentence a, b, e, or c refers to the precedent content closer than sentence f, we make a decision to put sentence d here. In this way, we get the final ordering: $a \succ b \succ e \succ c \succ d \succ f$.

I describe briefly how our ordering algorithm functions jointly with MDS. Let us reconsider the example shown in Figure 21. In this example, sentence extraction does not select sentence $c'$; sentence c is very similar to sentence $c'$. This may appear to be a rare case for explanation, but it could happen as we optimize a sentence-extraction method for MDS. The method described in Chapter 4 makes an effort to acquire information coverage under the condition that a number of sentences exist as summary candidates. This is to say that an extraction method should be capable of refusing redundant information.

When we collect articles that describe a series of events, we may find that lead sentences convey similar information throughout the articles because the major task of lead sentences is to give a subject. Therefore, it is quite natural that: lead sentences c and $c'$ refer to similar content; an extraction method for MDS does not choose both sentence $c'$ and c in terms of redundancy; and the method also prefers either sentence c or $c'$ in terms of information coverage.

### 5.5.2 *Implementation*

Figure 23 depicts a block diagram of the sentence ordering algorithm. Given nine sentences denoted by $\{a, b, ..., i\}$, the algorithm eventually produces an ordering: $a \succ b \succ f \succ c \succ i \succ g \succ d \succ h \succ e$.

We categorize sentences by their topics in the first phase. The aim of this phase is to group topically related sentences together. It was applied to sentence ordering by Barzilay et al. [7]. We use the vector space model [70] for sentence representation and apply the nearest neighbor method [15] to obtain topical clusters. Because sentences in newspaper articles are not always long enough to represent their

contents in sentence vectors, we assume that a newspaper article is written for one topic and thereby classify document vectors. Given $l$ articles and $m$ kinds of terms in the articles, we define a document-term matrix $D$ ($l \times m$), whose element $D_{ij}$ represents the frequency of term $j$ in document $i$,

$$D_{ij} = (\text{number of occurrences of term } j \text{ in document } i). \quad (5.10)$$

Letting $D_i$ denote a term vector ($i$-component row vector) of document $i$, we measure the distance or dissimilarity between two articles $x$ and $y$ using a cosine coefficient:

$$\text{distance}(D_x, D_y) = 1 - \frac{D_x \cdot D_y}{|D_x||D_y|}. \quad (5.11)$$

We apply the nearest neighbor method to merge a pair of articles when their minimum distance is lower than a given parameter $\alpha = 0.3$ (determined empirically). In this manner, we classify sentences according to topical clusters of articles. We determine an order of clusters based on the chronological order of the first publication date of articles in each cluster.

The rest phases of the algorithm, *chronological ordering* and *improving chronological ordering* that we described before, treat the partitioned sentences independently. We arrange sentences within respective topical clusters. In the Figure 23 example, we obtain two topical clusters, $\{a, b, c, f, g, i\}$ and $\{d, e, h\}$, as the output from the topical clustering. The second phase orders sentences in each topical group by the chronological order and sends two orderings, $a \succ b \succ c \succ i \succ g \succ f$ and $h \succ e \succ d$, to the third phase. The third phase refines each chronological ordering by the proposed method and outputs the final ordering: $a \succ b \succ f \succ c \succ i \succ g \succ d \succ h \succ e$.

### 5.5.3 *Experiment*

We conducted an experiment of sentence ordering through multi-document summarization to test the effectiveness of the proposed method. Performing an important sentence extraction by a method described in Chapter 4 up to the specified number of sentences (ca. 10% summarization rate), we produced a material for a summary, extracted sentences, for each task. We order the sentences by six methods: *human-made ordering (HO)* as the highest anchor; *random ordering (RO)* as the lowest anchor; *chronological ordering (CO)* as a conventional method; *chronological ordering with topical segmentation (COT)* (similar to Barzilay's method [7]); *the proposed method without topical segmentation (PO)*; and *the proposed method with topical segmentation (POT)*. Using 28 topics (summarization assignments) [1] in the TSC-3 test collection.

### 5.5.4 *Results*

Figure 24 shows distribution of rating scores of each method as a percentage of 84 ($28 \times 3$) summaries. Judges marked about 75% of

1 We exclude 2 of 30 summaries because they are so long (ca. 30 sentences) that it is hard for judges to evaluate and revise them.

Table 4. Comparison with human-made orderings

| | Spearman | | Kendall | | Continuity | |
|---|---|---|---|---|---|---|
| Method | AVG | SD | AVG | SD | AVG | SD |
| RO | -0.117 | 0.265 | -0.073 | 0.202 | 0.054 | 0.064 |
| CO | 0.838 | 0.185 | 0.778 | 0.198 | 0.578 | 0.218 |
| COT | 0.847 | 0.164 | 0.782 | 0.186 | 0.571 | 0.229 |
| PO | 0.843 | 0.180 | 0.792 | 0.184 | 0.606 | 0.225 |
| POT | 0.851 | 0.158 | 0.797 | 0.171 | 0.599 | 0.237 |
| HO | 1.000 | 0.000 | 1.000 | 0.000 | 1.000 | 0.000 |

human-made orderings (HOs) as either perfect or acceptable; they rejected as many as 95% of random orderings (ROs). Chronological ordering (CO) did not yield satisfactory results, losing a thread of 63% summaries, although CO performed much better than RO. Topical segmentation did not contribute to ordering improvement of CO either: COT was slightly worse than CO. After taking an in-depth look at the failure orderings, we found that topical clustering did not perform well during this test. We infer that topical clustering did not prove its merits with this test collection because the collection comprises relevant articles that were retrieved by some query and polished well by a human: they exclude articles that are unrelated to a topic. On the other hand, the proposed method (PO) improved chronological ordering much better than topical segmentation: the sum of the perfect and acceptable ratio jumped from 36% (CO) to 55% (PO). This fact shows that ordering refinement by precedence relation improves chronological ordering by pushing poor ordering to an acceptable level.

Table 4 shows the resemblance of orderings to those made by humans. Although we found that RO is clearly the worst, as in other results, we found no significant differences among CO, PO, and HO. This result revealed the difficulty of automatic evaluation by preparing a correct ordering.

Table 5 reports the resemblance of orderings to the corrected ones with average scores (AVG) and standard deviations (SD) of the three metrics $\tau_s$, $\tau_k$, and $\tau_c$. Apparently, average figures have a similar tendency to the rating task with three measures: HO is the best; PO is better than CO; and RO is definitely the worst. We applied one-way analysis of variance (ANOVA) to test the effect of these four different methods (RO, CO, PO, and HO). ANOVA verified the effects of the different methods ($p < 0.01$) for the three metrics. We also applied the Tukey test to compare the differences among these methods. The Tukey test revealed that RO was definitely the worst with all metrics. However, Spearman's rank correlation $\tau_S$ and Kendall's rank correlation $\tau_k$ failed to show significant differences among CO, PO, and HO. Only sentence continuity $\tau_c$ demonstrated that PO is superior to CO; and that HO is better than CO ($\alpha = 0.05$). The Tukey test proved that sentence continuity has better conformity to the rating results and higher discrimination to make a comparison.

Table 5. Comparison with corrected orderings

| | Spearman | | Kendall | | Continuity | |
|---|---|---|---|---|---|---|
| Method | AVG | SD | AVG | SD | AVG | SD |
| RO | 0.041 | 0.170 | 0.035 | 0.152 | 0.018 | 0.091 |
| CO | 0.838 | 0.185 | 0.870 | 0.270 | 0.775 | 0.210 |
| COT | 0.847 | 0.164 | 0.791 | 0.440 | 0.741 | 0.252 |
| PO | 0.843 | 0.180 | 0.921 | 0.144 | 0.856 | 0.180 |
| POT | 0.851 | 0.158 | 0.842 | 0.387 | 0.820 | 0.240 |
| HO | 0.949 | 0.157 | 0.947 | 0.138 | 0.922 | 0.138 |

As just described, the proposed method shows a significant improvement. However, evaluation by rating (Figure 24) and comparison with corrected ordering (Table 5) also present a great difference between PO and HO. The main reason they made such a difference is the way of arranging lead sentences. The proposed method is intended to preserve chronological order of lead sentences as long as the refinement algorithm does not choose them as a substitution of preceding information for an arranging sentence. A human can devise a presentation order from scratch without recognition of a lead sentence, but we did not consider preceding information of it, which is necessary to arrange a lead sentence.

In addition, several cases were found in which the proposed method inserted an unnecessary or inappropriate sentence as presuppositional information of a sentence. Because we do not apply a deep analysis of discourse structure and instead use precedent relation, a sentence does not always require all or any preceding sentences as presuppositional information. If the proposed method employs unnecessary preceding sentences as presuppositional information, it may choose a sentence that has little relation to the arranging sentence. The proposed method roughly estimates presuppositional information in this manner, but shows practical improvement for most summaries.

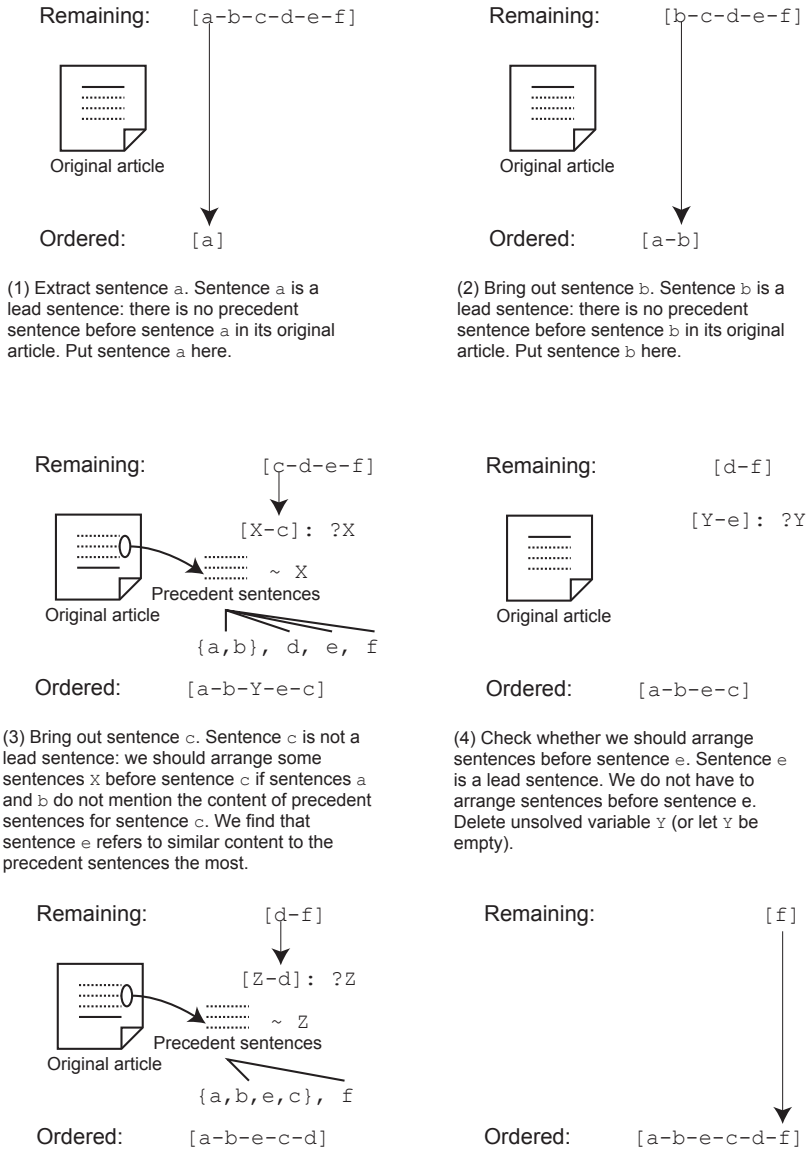



Figure 22. Improving chronological ordering using antecedent sentences

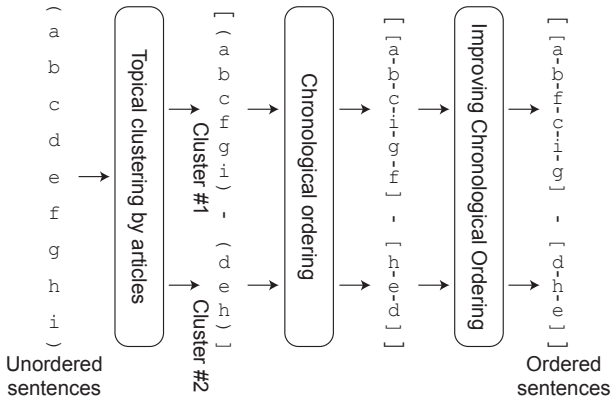



Figure 23. Outline of the ordering algorithm

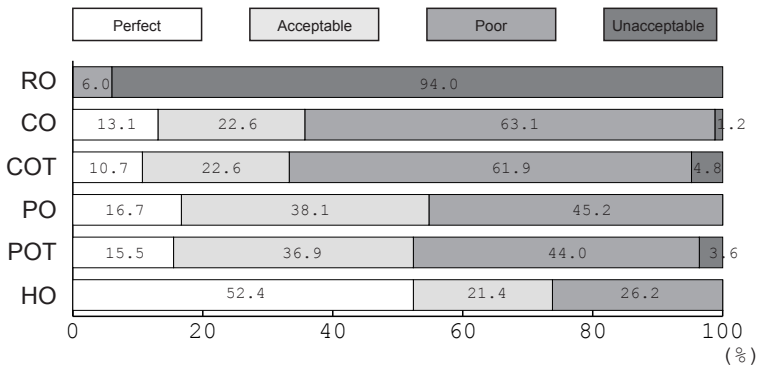



Figure 24. Distribution of the rating score of orderings (percent)

## 5.6 MACHINE-LEARNING APPROACH

### 5.6.1 *Bottom-up approach for text structuring*

The preceding section proposes a novel strategy for text structuring. Although several strategies to determine sentence ordering have been proposed, the appropriate way to combine these strategies to achieve more coherent summaries remains unresolved. As a collaborative study with my colleague Danushka Bollegala, the approach of the previous section was extended to formalize four criteria to capture the association of sentences in the context of multi-document summarization for newspaper articles. These criteria are integrated into a single criterion using a supervised learning approach. This section also proposes a bottom-up approach to arrange sentences; it repeatedly concatenates textual segments to obtain an overall segment with all sentences arranged.

We use the notation $a \succ b$ to represent that sentence $a$ precedes sentence $b$. We define term *segment* to describe a sequence of ordered sentences. When segment $A$ consists of sentences $a_1$, $a_2$, ..., $a_m$ in that order, we denote the situation as:

*Segment*

$$A = (a_1 \succ a_2 \succ ... \succ a_m). \tag{5.12}$$

Two segments $A$ and $B$ can be ordered as either $B$ after $A$ or $A$ after $B$. We define notation $A \succ B$ to represent that segment $A$ precedes segment $B$.

Let us consider a bottom-up approach to arrangement of sentences. Starting with a set of segments initialized with a sentence for each, we concatenate two segments into one segment that has the strongest association (discussed later) of all possible segment pairs. Repeating that concatenation will eventually yield a segment with all sentences arranged. The algorithm is considered as a variation of an agglomerative hierarchical clustering with the ordering information retained at each concatenating process.

The underlying idea of the algorithm, a bottom-up approach to text planning, was proposed by Marcu [47]. Assuming that semantic units (sentences) and their rhetorical relations (e.g., given that sentence *a* is an *elaboration* of sentence *d*) in a system, he transcribed a text structuring task into the problem of finding the best discourse tree that satisfies the set of rhetorical relations. He stated that global coherence can be achieved by satisfying local coherence constraints on ordering and clustering, thereby ensuring that the resultant discourse tree is well formed.

Unfortunately, identifying the rhetorical relation between two sentences has remained a difficult task for computers. Nevertheless, the bottom-up algorithm for arranging sentences can still be applied only if the association direction and strength of two segments (sentences) are defined. Hence, we introduce function $f(A \succ B)$ to represent the association direction and strength of segments $A$ and $B$,

*Association direction and strength*

$$f(A \succ B) = \begin{cases} p & \text{(if } A \text{ precedes } B\text{)} \\ 0 & \text{(if } B \text{ precedes } A\text{)} \end{cases}, \tag{5.13}$$

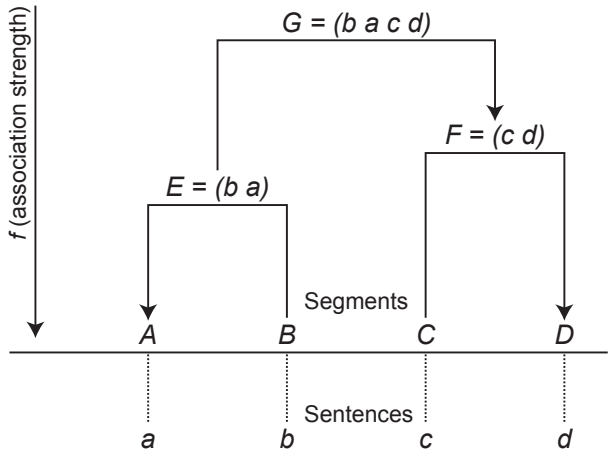



Figure 25. Arranging four sentences A, B, C, and D with a bottom-up approach

where $p$ ($0 \leqslant p \leqslant 1$) denotes the association strength of the segments A and B. The association strengths of two segments with different directions, e.g., $f(A \succ B)$ and $f(B \succ A)$, are not always identical in our definition, as

$$f(A \succ B) \neq f(B \succ A). \tag{5.14}$$

Figure 25 shows the process of arranging four sentences $a$, $b$, $c$, and $d$. First, we initialize four segments with a sentence for each,

$$A = (a), B = (b), C = (c), D = (d). \tag{5.15}$$

Presuming that $f(B \succ A)$ has the highest value of all possible pairs, e.g., $f(A \succ B)$, $f(C \succ D)$, we concatenate B and A to obtain a new segment:

$$E = (b \succ a). \tag{5.16}$$

Then we search for the segment pair with the strongest association. Supposing that $f(C \succ D)$ has the highest value, we concatenate C and D to obtain a new element:

$$F = (c \succ d). \tag{5.17}$$

Finally, comparing $f(E \succ F)$ and $f(F \succ E)$, we obtain the global sentence ordering of

$$G = (b \succ a \succ c \succ d). \tag{5.18}$$

*Four criteria for association*

In the above description, we have not defined the association of two segments. The previous work has addressed the association of textual segments (sentences) to obtain the coherent orderings. We define four criteria to capture the association of two segments: *chronology criterion*; *topical-closeness criterion*; *precedence criterion*; and *succession criterion*. These criteria are integrated into a function $f(A \succ B)$ using a machine-learning approach.

### 5.6.2 *Criteria for arranging sentences*

*Chronology criterion*

*Chronology criterion* reflects the chronological ordering [37, 50], which arranges sentences in a chronological order of the publication date. We define the association strength of arranging segments $B$ after $A$ measured using chronology criterion $f_{chro}(A \succ B)$ as the following formula.

$$f_{chro}(A \succ B) = \begin{cases} 1 & T(a_m) < T(b_1) \\ 1 & [D(a_m) = D(b_1)] \wedge [N(a_m) < N(b_1)] \\ 0.5 & [T(a_m) = T(b_1)] \wedge [D(a_m) \neq D(b_1)] \\ 0 & \text{otherwise} \end{cases} \tag{5.19}$$

Therein: $a_m$ represents the last sentence in segment $A$; $b_1$ represents the first sentence in segment $B$; $T(s)$ is the publication date of sentence $s$; $D(s)$ is the unique identifier of the document to which sentence $s$ belongs: and $N(s)$ denotes the line number of sentence $s$ in the original document. The chronological order of arranging segment $B$ after $A$ is determined using the comparison between the last sentence in the segment $A$ and the first sentence in the segment $B$.

The chronology criterion assesses the appropriateness of arranging segment $B$ after $A$ if: sentence $a_m$ is published earlier than $b_1$; or sentence $a_m$ appears before $b_1$ in the same article. The criterion assumes the order to be undefined if sentences $a_m$ and $b_1$ are published on the same day, but appear in different articles. If none of the above conditions are satisfied, the criterion estimates segment $B$ to precede $A$.

*Topical-closeness criterion*

The topical-closeness criterion deals with the association based on the topical similarity of two segments. The criterion reflects the ordering strategy proposed by Barzilay et al. [7], which groups the sentences that refer to the same topic. To measure topical closeness of two sentences, we represent each sentence with a vector whose elements correspond to the occurrence of the nouns and verbs in the sentence. The vector values are represented as Boolean values: 1 if the sentence contains the word, otherwise 0.

We define the topical closeness of two segments $A$ and $B$ as

$$f_{topic}(A \succ B) = \frac{1}{|B|} \sum_{b \in B} \max_{a \in A} \text{sim}(a, b). \tag{5.20}$$

Therein, $\text{sim}(a, b)$ denotes the similarity of sentences $a$ and $b$ calculated using the cosine similarity of two vectors corresponding to sentences $a$ and $b$. For sentence $b \in B$, $\max_{a \in A} \text{sim}(a, b)$ chooses the most similar sentence $a \in A$ to sentence $b$ and yields the similarity. The topical-closeness criterion $f_{topic}(A \succ B)$ assigns a higher value when the topic referred by segment $B$ is the same as that of segment $A$.

We consider the case in which we arrange segment $A$ before $B$. Each sentence in segment $B$ has the presuppositional information that should be conveyed to a reader in advance. Given sentence $b \in B$,

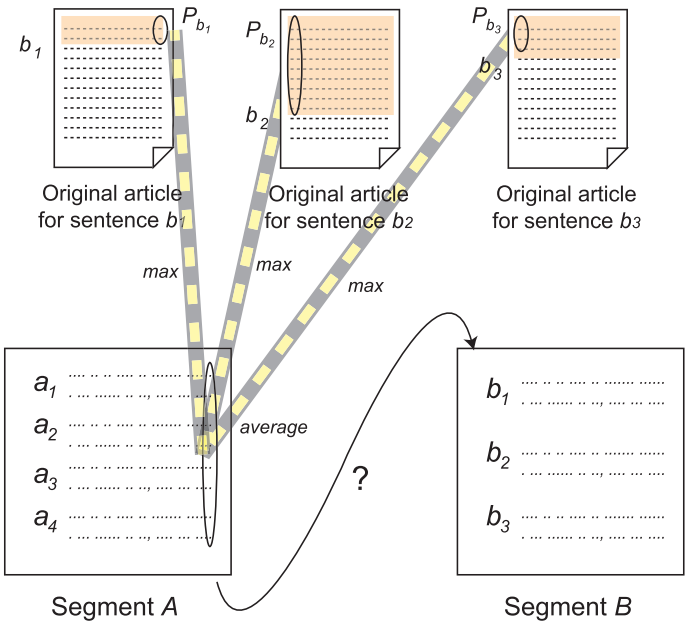



Figure 26. Precedence criterion

such presuppositional information might be presented by the sentences appearing before sentence $b$ in the original article. However, we cannot guarantee whether a sentence-extraction method for multi-document summarization chooses any sentences before $b$ for a summary because the extraction method usually determines a set of sentences within the constraint of summary length that maximize information coverage and exclude redundant information. The *precedence criterion* measures the substitutability of the presuppositional information of segment $B$ (e.g., the sentences appearing before sentence $b$) as segment $A$. This criterion is a formalization of the sentence-ordering algorithm proposed in Section 5.5.

*Precedence criterion*

We define the precedence criterion as the following formula:

$$f_{pre}(A \succ B) = \frac{1}{|B|} \sum_{b \in B} \max_{a \in A, p \in P_b} sim(a, p). \tag{5.21}$$

Therein, $P_b$ is a set of sentences appearing before sentence $b$ in the original article, and $sim(a, b)$ denotes the cosine similarity of sentences $a$ and $b$ (defined just as in the topical-closeness criterion). Figure 26 shows an example of calculating the precedence criterion for arranging segment $B$ after $A$. We approximate presuppositional information for sentence $b$ by sentences $P_b$, i.e., sentences appearing before the sentence $b$ in the original article. Calculating the similarity among sentences in $P_b$ and $A$ using the maximum similarity of the possible sentence combinations, Formula 5.21 is interpreted as the average similarity of the precedent sentences $P_b (b \in B)$ to the segment $A$.

*Succession criterion*

The idea of a *succession criterion* is exactly opposite from the precedence criterion. Given that we arrange segments $B$ after $A$, the suc-

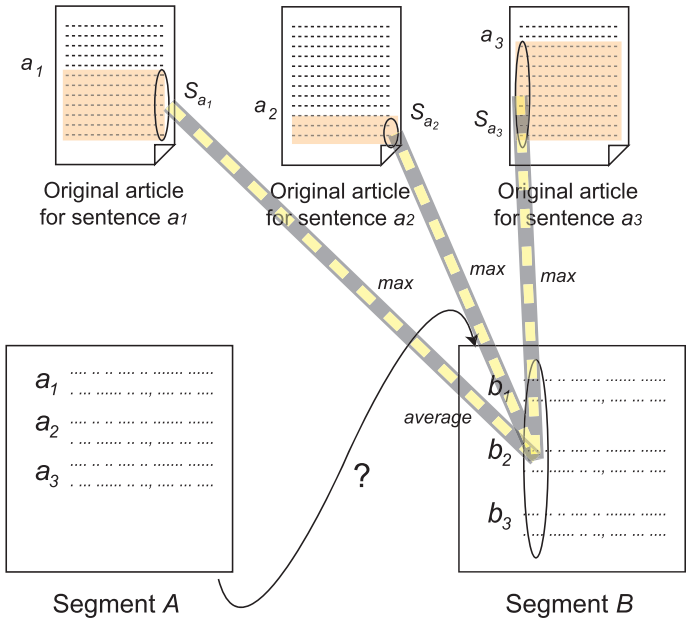



Figure 27. Succession criterion

cession criterion assesses how much the succeeding information for segment A is covered by segment B:

$$f_{succ}(A \succ B) = \frac{1}{|A|} \sum_{a \in A} \max_{s \in S_a, b \in B} sim(s, b). \tag{5.22}$$

Therein, $S_a$ is a set of sentences appearing after sentence $a$ in the original article, and $sim(a, b)$ denotes the cosine similarity of sentences $a$ and $b$ (defined merely as the topical-closeness criterion). Figure 27 shows an example of calculating the succession criterion to arrange segments B after A. The succession criterion measures the substitutability of the succeeding information (e.g., the sentences appearing after the sentence $a \in A$) as segment B.

### 5.6.3 *SVM classifier to assess the integrated criterion*

We integrate the four criteria described above to define the function $f(A \succ B)$ to represent the association direction and strength of the two segments A and B (Formula 5.13). More specifically, given the two segments A and B, function $f(A \succ B)$ is defined to yield the integrated association strength from four values: $f_{chro}(A \succ B)$, $f_{topic}(A \succ B)$, $f_{pre}(A \succ B)$, and $f_{succ}(A \succ B)$. We formalize the integration task as a binary classification problem and employ a support vector machine (SVM) as the classifier. We conducted supervised learning as follows.

*Preparing training data*

We partition a human-ordered extract into pairs, each of which consists of two non-overlapping segments. Let us explain the partitioning process taking four human-ordered sentences, $a \succ b \succ c \succ d$ shown in Fig. 28. First, we place the partitioning point immediately

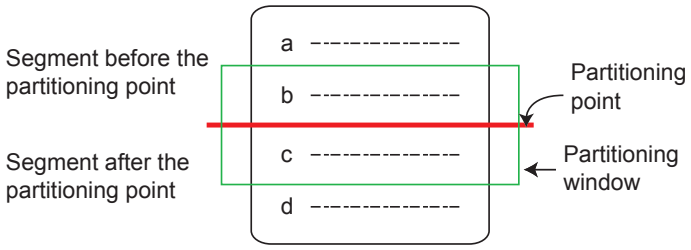



Figure 28. Partitioning a human-ordered extract into pairs of segments

$$+1 : [f_{chro}(A \succ B), f_{topic}(A \succ B), f_{pre}(A \succ B), f_{succ}(A \succ B)]$$
$$-1 : [f_{chro}(B \succ A), f_{topic}(B \succ A), f_{pre}(B \succ A), f_{succ}(B \succ A)]$$

Figure 29. Two vectors in a training data generated from two ordered segments $A \succ B$

after the first sentence $a$. Specifically addressing sentence $a$ arranged immediately before the partition point and sentence $b$ arranged immediately after, we identify the pair $\{(a), (b)\}$ of two segments $(a)$ and $(b)$. Enumerating all possible pairs of two segments facing immediately before/after the partitioning point, we obtain the following pairs, $\{(a), (b \succ c)\}$ and $\{(a), (b \succ c \succ d)\}$. Similarly, segment pairs, $\{(b), (c)\}$, $\{(a \succ b), (c)\}$, $\{(b), (c \succ d)\}$, $\{(a \succ b), (c \succ d)\}$, are obtained from the partitioning point between sentences $b$ and $c$. Collecting the segment pairs from the partitioning point between sentences $c$ and $d$ (i.e., $\{(c), (d)\}$, $\{(b \succ c), (d)\}$ and $\{(a \succ b \succ c), (d)\}$), we identify ten pairs in all from the four ordered sentences. In general, this process yields $n(n^2 - 1)/6$ pairs from $n$ ordered sentences. From each pair of segments, we generate one positive and one negative training instance as follows.

Given a pair of two segments $A$ and $B$ arranged in an order $A \succ B$, we calculate four values, $f_{chro}(A \succ B)$, $f_{topic}(A \succ B)$, $f_{pre}(A \succ B)$, and $f_{succ}(A \succ B)$ to obtain the instance with the four-dimensional vector (Fig. 29). We label the instance (corresponding to $A \succ B$) as a positive class (i.e., $+1$). Simultaneously, we obtain another instance with a four-dimensional vector corresponding to $B \succ A$. We label it as a negative class (i.e., $-1$). Accumulating these instances as training data, we obtain a binary classifier using an SVM with a quadratic kernel. The SVM classifier yields the association direction of two segments (e.g., $A \succ B$ or $B \succ A$) with the class information (i.e., $+1$ or $-1$). We assign the association strength of two segments using the class probability estimate that the instance belongs to a positive ($+1$) class. We set the association strength as zero when an instance is classified into a negative ($-1$) class (see the definition of formula 5.13).

Table 6. Correlation between two sets of human-ordered extracts

| Metric | Mean | S td. Dev | Min | Max |
|---|---|---|---|---|
| Spearman | 0.739 | 0.304 | -0.2 | 1 |
| Kendall | 0.694 | 0.290 | 0 | 1 |
| Average Continuity | 0.401 | 0.404 | 0.001 | 1 |

### 5.6.4 *Evaluation*

We evaluated the proposed method using the Third Text Summarization Challenge (TSC-3) corpus[2]. The TSC-3 corpus contains 30 sets of extracts, each of which consists of unordered sentences extracted from Japanese newspaper articles relevant to a topic (query). Each extract comprises ca. 15 sentences. We arrange the extracts using different algorithms and evaluate the readability of the ordered extracts using subjective grading and semi-automatic evaluation.

To construct a training dataset applicable to the proposed method, we asked two human subjects to arrange the extracts and obtained $30(\text{topics}) \times 2(\text{humans}) = 60$ sets of ordered extracts. Table 6 shows the agreement of the ordered extracts between the two subjects. The correlation is measured using three metrics: Spearman's rank correlation, Kendall's rank correlation, and average continuity (described later). The mean correlation values (0.74 for Spearman's rank correlation and 0.69 for Kendall's rank correlation) indicate a certain level of agreement in sentence orderings made using the two subjects. In fact, 8 out of 30 extracts were identical.

We applied the leave-one-out method for the proposed method to produce a set of sentence orderings. In this experiment, the leave-one-out method arranges an extract using an SVM model trained from the remainder of the 29 extracts. Repeating this process 30 times with a different topic for each iteration, we generated a set of 30 extracts for evaluation. In addition to the proposed method, we prepared seven sets of sentence orderings produced by different algorithms for comparison. We describe briefly the eight algorithms (including the proposed method).

AGGLOMERATIVE ORDERING (AGL) is an ordering arranged using the proposed method.

RANDOM ORDERING (RND) is an ordering as the lowest anchor in which sentences are arranged randomly.

HUMAN-MADE ORDERING (HUM) is the highest anchor in which sentences are arranged by a human subject.

PROBABILISTIC ORDERING (PRO) arranges sentences using the probabilistic text structuring method proposed by Lapata [35]. We used CaboCha[3] (a parser for Japanese text) to obtain part-of-speech information and dependency structure of sentences. Using

2 TSC-3 Website: http://lr-www.pi.titech.ac.jp/tsc/tsc3-en.html
3 http://chasen.org/ taku/software/cabocha/

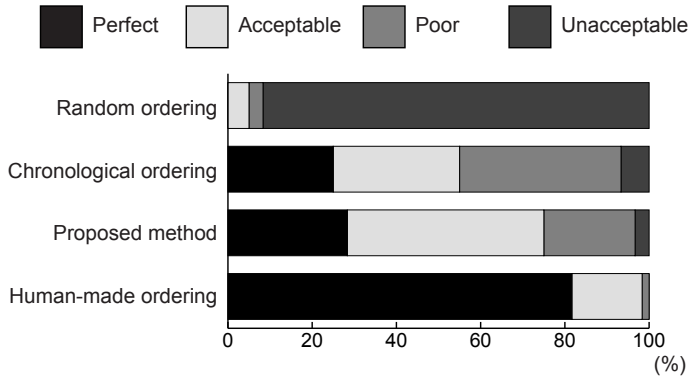



Figure 30. Subjective grading

nouns, verbs, and verb-noun dependencies, we trained the language model with a corpus of 100,000 articles in Mainichi and Yomiuri news papers.

CHRONOLOGICAL ORDERING (CHR) arranges sentences with the chronology criterion defined in Formula 5.19. Sentences are arranged in the chronological order of their publication date.

TOPICAL-CLOSENESS ORDERING (TOP) arranges sentences with the topical-closeness criterion defined in Formula 5.20.

PRECEDENCE ORDERING (PRE) arranges sentences with the precedence criterion defined in Formula 5.21.

SUCEEDENCE ORDERING (SUC) arranges sentences with the succession criterion defined in Formula 5.22.

The last four algorithms (CHR, TOP, PRE, and SUC) arrange sentences solely according to the corresponding criteria, each of which uses the association strength directly to arrange sentences without the integration of other criteria. These orderings are expected to reflect the performance of each expert independently and to reflect their respective contributions to solve the sentence-ordering problem.

### 5.6.5 *Results*

Figure 30 shows the distribution of the subjective grading made by two judges to four sets of orderings, RND, PRO, CHR, and AGL. Each set of orderings has 30(topics) $\times$ 2(judges) $=$ 60 ratings. Most RND orderings are rated as *unacceptable*. Although CHR and AGL orderings have roughly the same number of *perfect* orderings (ca. 25%), the AGL algorithm gained more *acceptable* orderings (47%) than the CHR algorithm (30%). This fact shows that integration of CHR experts with other experts worked well by pushing poor ordering to an acceptable level. However, a huge gap between *AGL* and *HUM* orderings was also apparent. The judges rated 28% AGL orderings as *perfect*, but the figure jumped to as many as 82% for HUM orderings.

Table 7. Comparison to human-made ordering

| Method | Spearman coefficient | Kendall coefficient | Average Continuity |
|---|---|---|---|
| RND | -0.127 | -0.069 | 0.011 |
| PRO | 0.076 | 0.068 | 0.037 |
| TOP | 0.414 | 0.400 | 0.197 |
| PRE | 0.415 | 0.428 | 0.293 |
| SUC | 0.473 | 0.476 | 0.291 |
| CHR | 0.583 | 0.587 | 0.356 |
| AGL | 0.603 | 0.612 | 0.459 |

We also evaluate sentence orderings by reusing two sets of gold-standard ordering made for the training data. In general, the subjective grading consumes much time and effort, even though we cannot reproduce the evaluation afterward. Precedent studies [7, 35] have used rank correlation coefficients such as Spearman's rank correlation and Kendall's rank correlation, assuming a sentence ordering to be a rank. Section 5.3 proposed a metric that assesses continuity of pairwise sentences compared with the gold standard. In addition to Spearman's and Kendall's rank correlations, we proposed an *average continuity* metric, which extends the idea of the continuity metric to continuous $k$ sentences.

Table 7 reports the resemblance of orderings produced by seven algorithms to the human-made ones with three metrics, Spearman's rank correlation, Kendall's rank correlation, and Average Continuity. The proposed method (AGL) outperforms the rest in all evaluation metrics, but the chronological ordering (CHR) appeared to play the major role. The one-way analysis of variance (ANOVA) verified the effects of different algorithms for sentence orderings with all metrics ($p < 0.01$). We used Tukey's Honest Significant Differences (HSD) test to compare differences among these algorithms. The Tukey test revealed that AGL was significantly better than the rest. Also, PRO performed poorly in our experiments. Lapata's model assumes that the position of a sentence in a summary depends only upon the sentence that directly precedes it. However, a careful review of human-ordered extracts reveals that the scope of the dependency of a sentence goes well beyond its direct precedent. On the other hand, TOP, PRE and SUC ordering heuristics operate upon blocks of sentences and reports higher Spearman, Kendall and average continuity values than does PRO.

## 5.7 SUMMARY

This chapter addressed a drawback of chronological ordering, which is widely used by conventional summarization systems: it arranges sentences without considering presupposed information of each sentence. Proposing a method to improve chronological ordering by resolving

precedent information of arranging sentences, I conducted an experiment of sentence ordering through MDS. We also proposed an evaluation metric that measures sentence continuity and an amendment-based evaluation task. The proposed method, which uses the precedence relations of sentences, achieved good results, raising poor chronological orderings to an acceptable level by 20%. Amendment-based evaluation outperformed an evaluation that compares an ordering with an answer made by a human. The sentence continuity metric, when applied to the amendment-based task, showed good agreement with the rating result.

This chapter also presents a bottom-up approach to arrange sentences extracted for multi-document summarization. Different strategies to arrange sentences are integrated into a single framework with a supervised learning approach. Our experimental results with sentences extracted from multiple newspaper articles showed a significant improvement over existing sentence-ordering strategies. However, the results also implied that chronological ordering played a major role in arranging sentences. A future direction of this study would be to explore application of the proposed framework to more generic texts, i.e., a document collection without chronological information.

# Part III

# COMPACTION

# 6 ABBREVIATION

## 6.1 INTRODUCTION

*Abbreviations as a term variation*

Abbreviations result from a highly productive type of term variation that substitutes fully expanded terms (e.g. *European Union*) with shortened term-forms (e.g. *EU*). Among biomedical studies, Chang and Schütze [14] reported that 64,242 new abbreviations were introduced in 2004 in MEDLINE abstracts. Terminological resources and scientific databases for biomedical literature (such as UMLS[1], Swiss-Prot[2], SGD[3], FlyBase[4], and UniProt[5]) cannot maintain up-to-date information to match the growth of neologisms [63]. Wren et al. [78] reported that, with the MEDLINE database, 5,477 documents were retrieved using the abbreviation *JNK* while only 3,773 documents were retrievable using its full term, *c-jun N-terminal kinase*.

Abbreviations also hinder automatic text summarization of newspaper articles. Let us take an example of two articles about a bidding war between Nippon Telegraph and Telephone (NTT) and Cable and Wireless (C&W) in 1999.

> *Cable and Wireless (C&W) is stepping up its battle to win control of Japanese phone group International Digital Communications (IDC). It is facing a titanic struggle with NTT, Japan's largest phone company, to win control of IDC. C&W plans to up its bid for IDC to 344 m GBP on Friday, as competition to grab a stake in Japan's deregulating phone business gathers pace.*
>
> — BBC News (excerpt), Thursday, May 6, 1999.

> *Cable and Wireless (C&W) raised its bid for Japanese phone group International Digital Communications (IDC). Its move came a day after rival bidder and Japanese market leader, Nippon Telegraph and Telephone (NTT), upped its own offer for IDC. C&W launched the cross-border takeover battle, unprecedented in Japan, last month with a 62.4 bn yen (321 m GBP) bid. The raised offer – the third – now values IDC at 69 bn yen (356 m GBP).*
>
> — BBC News (excerpt), Tuesday, June 1, 1999.

*Normalizing abbreviations*

These sentences contain three abbreviations with their expanded form, *Cable and Wireless (C&W)*, *Nippon Telegraph and Telephone (NTT)*, and *International Digital Communications (IDC)*. We would deem the summary redundant because the three abbreviations would be defined multiple times if these sentences yielded a summary from multiple articles. An ideal solution for a summarization system to deal with

1 http://www.nlm.nih.gov/research/umls/
2 http://www.ebi.ac.uk/swissprot/
3 http://www.yeastgenome.org/
4 http://www.flybase.org/
5 http://www.ebi.ac.uk/GOA/

these abbreviations is:

1. To recognize abbreviations used in the source documents.
2. To define an abbreviation (i.e. abbreviation with its full form) at its first occurrence in a summary.
3. To use the shortened form of the abbreviation in its latter occurrences.

Presuming that we used the first sentence in each article to produce an extract, a summary would have normalized abbreviations:

> *Cable and Wireless (C&W) is stepping up its battle to win control of Japanese phone group International Digital Communications (IDC). C&W has raised its bid for Japanese phone group IDC.*

Note that the second sentence uses only the shortened forms: *C&W* and *IDC*.

A salient challenge of text mining is dealing with an enormous amount of documents in a scalable and efficient manner. Simultaneously, we can utilize the amount of textual data to obtain accurate and comprehensive results. We present a methodology for building a good quality abbreviation dictionary of common abbreviations and their expanded forms, making effective use of large amounts of text. This chapter is organized as follows. The subsequent section (Section 6.2) reviews the previous work on abbreviation extraction. Section 6.3 presents a methodology for abbreviation recognition based on statistical information in a large corpus. Section 6.4 reports the evaluation results on English test collection.

Even though the statistical approach is independent from the target language, the creation process of Japanese abbreviations is too complicated to deal with the same method for English abbreviations. Section 6.5 examines the usages of parenthetical expressions in Japanese newspaper articles. The subsequent section (Section 6.6) presents a method to classify parenthetical expressions into paraphrasable and non-paraphrasable groups. Section 6.7 reports experiments on Japanese text collection. I conclude this chapter in Section 6.8 with the summary of the outcomes.

## 6.2 RELATED WORK

*Global and local abbreviations*

Gaudan et al. [22] distinguished *global* abbreviations from *local* abbreviations based on the presence of their definitions in texts. Global abbreviations appear in documents without the expanded form explicitly stated, while local abbreviations accompany their expanded forms in the document. Global abbreviations hinder text-mining tasks such as information retrieval and information extraction, appearing in text without explicit definitions of their expanded forms.

Thus, an *abbreviation dictionary* is necessary for advanced text-mining tasks to establish associations between abbreviations and their expanded forms. Adar [1] noted that previous work had mostly found abbreviation definitions within a text in a similar manner to information extraction. He saw the need for additional tasks for a practical

abbreviation resource such as merging similar definitions and providing disambiguation information. Although we find such components indispensable for text-mining applications, here we focus on the task of finding abbreviation definitions, as the first step in building an *accurate* abbreviation dictionary.

*Dynamic and common abbreviations*

Another important aspect for building an abbreviation dictionary is the distinction between *dynamic* and *common* abbreviations [80]. Dynamic abbreviations are one-time substitutions valid within a document and therefore always local. In contrast, common abbreviations are used over two or more publications, and may appear in documents with or without their expanded forms. An abbreviation dictionary should focus on common abbreviations since they are potential global abbreviations, i.e., might be written without their definitions in some documents. This study does not deal with the identification of dynamic abbreviations which can be recognized by letter matching techniques, but collects definitions of local and common abbreviations in source documents.

*Abbreviation recognition*

In practice, no generic rules or exact patterns have been established for dealing with abbreviation creation. Thus, abbreviation recognition aims to extract pairs of short forms (acronyms or abbreviations) and long forms (their expanded forms or definitions) occurring in text. Except for a few studies (e.g. Sakai and Masuyama [68]), most studies share pattern (6.1) to locate a textual fragment with an abbreviation and its expanded form [73, 77].

*Parenthetical expression*

$$\textit{long form}\ \text{'('}\ \textit{short form}\ \text{')'} \tag{6.1}$$

For example, the sentence, "The exact *route was determined by magnetic resonance imaging (MRI)*", could yield the textual fragment marked with the italic letters, assuming that we take $(l+4)$ words appearing before the parenthetical expression [1], where $l$ is the number of letters in the short form. The task is to identify the "authentic" long-form in the textual fragment if any. Existing methods for solving this problem can be categorized into three groups: using heuristics and/or scoring rules [1, 2, 73, 75, 77, 80]; machine learning [14, 52, 59]; and statistics [27, 39].

*Classification of existing methods*

The first category uses predefined heuristic rules/algorithms to find a long form in a textual fragment. For example, Schwartz and Hearst [73] implemented a letter-matching algorithm that maps all alpha-numerical letters in the short form to the long form, starting from the end of both the short and long forms and moving right to left. Even though the core algorithm is very simple, the authors report 96% precision and 82% recall on the Medstract gold standard[6]. Adar [1] proposes scoring rules to find the most likely long-form, accepting multiple long-form candidates, e.g., *determined by magnetic resonance imaging (MRI)* and *magnetic resonance imaging (MRI)* in the fragment, yielding 95% precision and 85% recall on the Medstract corpus.

*Letter-matching algorithm*

*Medstract gold standard*

*Scoring rules*

The second category obtains such rules by using a machine learning technique. Chang and Schütze [14] applied a logistic regression to calculate the likelihood of long-form candidates. They enumerate possible long-form candidates with Longest Common Substring (LCS) formalization [75]. The likelihood of the candidates is estimated as the

*Logistic regression*

*Longest Common Substring (LCS)*

6 http://www.medstract.org/

probability calculated from a logistic regression with nine features such as the percentage of long-form letters aligned at the beginning of a word, the percentage of short-form letters aligned to the long form, etc. Their method achieved 80% precision and 83% recall on the Medstract corpus.

The third category utilizes statistical clues in the source documents, e.g., co-occurrence between short forms and long forms. Hisamitsu and Niwa [27] proposed a method for extracting useful parenthetical expressions from Japanese newspaper articles. Their method measures the co-occurrence strength between the inner and outer phrases of a parenthetical expression via mutual information, $\chi^2$ test with Yate's correction, Dice coefficient, log-likelihood ratio, etc. Unfortunately, their method deals with generic parenthetical expressions (i.e., abbreviation, non-abbreviation paraphrases, supplementary comments), not focusing exclusively on abbreviation recognition.

*Co-occurrence strength*

Liu and Friedman [39] based their method on collocations occurring before the parenthetical expressions. Enumerating long-form candidates as collocations appearing more than once in a text collection, their method eliminates unlikely candidates with rules such as "remove a set of candidates $T_w$ formed by adding a prefix word to a candidate $w$ if the number of such candidates $T_w$ is greater than 3"; "remove a candidate $t \in T_w$ if the occurrence frequency of candidate $t$ divided by the occurrence frequency of candidate $w$ is smaller than 0.25"; and "remove a candidate $w$ if the summation of occurrence frequency of $T_w$ divided by the occurrence frequency of candidate $w$ is greater than 0.9". They report a precision of 96.3% and a recall of 88.5% for abbreviation recognition on their test corpus.

*Collocation mining*

In addition to abbreviation recognition, some studies aimed at modeling the process of acronym generation. Tsuruoka et al. [76] modeled the process of generating English acronyms as a sequence labeling problem. Defining actions to generate acronyms from expanded forms with five operations, *skip*, *upper*, *lower*, *space*, and *hyphen*, they applied the Maximum Entropy Markov Model (MEMM) [10] to obtain discrimination probability between an expanded form and its acronyms. Murayama and Okumura [51] also modeled the process of generating Japanese acronyms by using a noisy-channel model [13, 17]. However, it is difficult for the generation approach to achieve a high accuracy because the process of abbreviation generation is complicated. I will address the generation processes of Japanese abbreviations in Section 6.5.

*Maximum Entropy Markov Model (MEMM)*

*Noisy-channel model*

## 6.3 EXTRACTING ENGLISH ABBREVIATIONS

### 6.3.1 *Recognizing abbreviations based on co-occurrence*

We assume a word sequence is a possible long-form if the word sequence co-occurs frequently with a specific abbreviation and not with other surrounding words. A sequence of words that co-occurs with an abbreviation does not always imply the abbreviation-definition relation: the abbreviation *5-HT* co-occurs frequently with the term *serotonin*, but their relation is interpreted as a synonymous relation. We deal with this issue with a validation rule (described later). Satisfying the

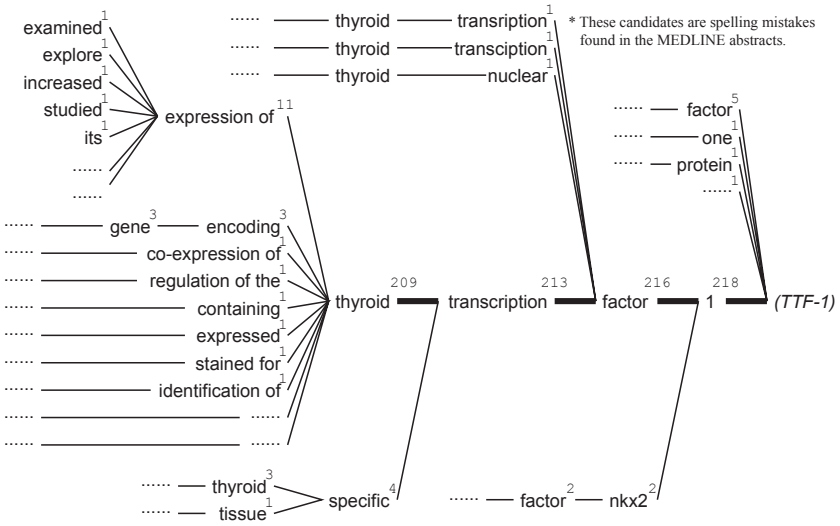



Figure 31. Expressions appearing before the abbreviation *TTF-1* in parentheses.

validation rule for being a long form, the word sequence is stored in the abbreviation dictionary. Figure 31 illustrates our assumption with the abbreviation *TTF-1*. The tree consists of expressions collected from all sentences with the abbreviation *TTF-1* in parentheses and appearing before the abbreviation. A node represents a word, and a path from any node to *TTF-1* represents a long-form candidate. The words with function words (e.g., *expression of*, *regulation of the*, etc.) are merged into a node. This is due to the requirement for a long-form candidate discussed later (Section 6.3.2). The figure above each node shows the co-occurrence frequency of the corresponding long-form candidate. For example, long-form candidates *1*, *factor 1*, *transcription factor 1*, and *thyroid transcription factor 1* co-occur 218, 216, 213, and 209 times respectively with the abbreviation *TTF-1* in the text collection.

*Expressions before abbreviation TTF-1*

Even though long-form candidates *1*, *factor 1* and *transcription factor 1* co-occur frequently with *TTF-1*, they also co-occur frequently with *thyroid*. Meanwhile, the candidate *thyroid transcription factor 1* is used in a number of contexts (e.g., *expression of thyroid transcription factor 1*, *expressed thyroid transcription factor 1*, etc). Therefore, we observe the strongest relationship is between abbreviation *TTF-1* and its long-form candidate *thyroid transcription factor 1* in the tree. We apply a validation rule (described later) to the long-form candidate to make sure an abbreviation-definition relation does occur. In this example, the candidate pair is likely to be in an abbreviation-definition relation as the long form *thyroid transcription factor 1* contains all the alphanumeric letters in the short form *TTF-1*.

*Long-form detection based on co-occurrence*

This approach detects the starting point of the long form without using letter matching. A simple method based on letter matching may misrecognize the long form *transcription factor 1* since it also contains the necessary elements to produce the abbreviation *TTF-1*. Whereas previous work dealt with this case by introducing, e.g., a set of com-

plicated rules, scoring, or machine-learning techniques, our approach uses overlapping definitions of an abbreviation stated by a number of authors. This characteristic of our approach also contributes to finding a long form whose short form is arranged in a different word order such as *beta 2 adrenergic receptor (ADRB2)* and *water activity (AW)*.

### 6.3.2 *Term recognition approach to long-form recognition*

*Long-form recognition as a term recognition*

Having collected all sentences with a specific abbreviation (hereafter *contextual sentences*), we deal with the problem of extracting long-form candidates from the contextual sentences in a similar manner to the term recognition task which extracts terms from a given text. For this purpose, we modified the C-value method [21], a domain-independent method for automatic term recognition (ATR). The C-value approach is characterized by the extraction of *nested* terms that gives preference to terms appearing frequently in a given text but not as a part of specific longer terms. This is a desirable feature for us as we wish to recognize word sequences co-occurring frequently with a specific abbreviation and not with other surrounding words.

*C-value method*

The C-value method combines linguistic and primarily statistical information. Linguistic analysis enumerates all possible terms in a given text by applying part-of-speech tagging, candidate extraction, and a stop-list. For example, pattern[7] (6.2) has been used for extracting term candidates:

```
[:ADJ:]*[:NOUN:]+
```
(6.2)

We know the position to search for a long form in a contextual sentence, i.e., a word or word sequence just before an abbreviation in parentheses. As it is preferable for long-form recognition to be independent of the terminological knowledge for the target domain, we omit part-of-speech tagging.

Given a contextual sentence, we tokenize it by non-alphanumeric characters (e.g., space, hyphen, colon) and apply a stemming algorithm [62] to obtain a sequence of normalized words. Pattern (6.3)[8] extracts long-form candidates from the sequence:

*Extraction pattern*

```
[:WORD:].*$
```
(6.3)

The extraction pattern accepts a word or word sequence if it begins with any non-function word[9], and ends with any word just before the corresponding short form in the contextual sentence.

Consider the example of a contextual sentence, "we studied the expression of thyroid transcription factor-1 (TTF-1)". We extract the following substrings as long-form candidates (words are stemmed): *1*; *factor 1*; *transcript factor 1*; *thyroid transcript factor 1*; *expression of thyroid transcript factor 1*; and *studi the expression of thyroid transcript factor 1*. Substrings such as *of thyroid transcript factor 1* (which begins

7 `[:ADJ:]` and `[:NOUN:]` match an adjective and noun respectively.

8 `[:WORD:]` matches a non-function word; `.*` matches an empty string or any word(s) of any length; and `$` matches a short form of the target abbreviation.

9 29 function words are held in an external dictionary: three articles (*a*, *an*, *the*); two conjunctions (*and*, *or*); seventeen prepositions (*of*, *to*, *in*, etc); seven forms of the verb *be*.

with a function word) and *thyroid transcript* (which ends prematurely before the short form) are not selected as long-form candidates. The list of function words is not used for removing specific words in long-form candidates (e.g., *expression of thyroid transcript factor 1* contains a function word *of*), but for preventing invalid candidates beginning with a function word such as *of thyroid transcript factor 1*.

The original C-value method assigns a termhood (likelihood to be a term) to a candidate term by using the features: frequency of occurrence of the candidate term; frequency of the candidate term as part of other longer candidate terms; number of these longer candidate terms; and length of the candidate term. The original termhood function $CV(c)$ is defined in formula formula 6.4,

*Definition of the C-value method*

$$CV(c) = \log[len(c)] \cdot freq(c) - \frac{\sum_{t \in T_c} freq(t)}{|T_c|}. \tag{6.4}$$

In formula 6.4, $c$ is a candidate term; $freq(c)$ denotes the frequency of occurrence of term $c$; $len(c)$ denotes the length (number of words) of term $c$; $T_c$ is a set of candidate terms which contain term $c$; $t \in T_c$ is a candidate term which contains term $c$; and $|T_c|$ represents the number of such candidate terms $T_c$. Multiplying $\log[len(c)]$ with $freq(c)$ is based on the consideration that a longer string appears less frequently than a shorter string [21]. This is preferred for calculating termhood for term candidates extracted by part-of-speech information. However, longer terms are not useful as long forms, as the previous work excluded candidates longer than the maximum length estimated by the number of letters in a short form [61]. In addition, formula 6.4 always yields zero for a one-word candidate.

*Problem of the C-value method*

Formula 6.5 amends the original formula of C-value (formula 6.4) to define the *long-form likelihood* $LH(c)$ for a candidate $c$:

*Long-form likelihood*

$$LH(c) = freq(c) - \sum_{t \in T_c} freq(t) \times \frac{freq(t)}{\sum_{t \in T_c} freq(t)}. \tag{6.5}$$

In formula 6.5, $c$ is a long-form candidate; $freq(c)$ denotes the frequency of occurrence of a candidate $c$ in the contextual sentences (i.e., co-occurrence frequency with a short form); and $T_c$ is a set of nested long-form candidates, each of which consists of a preceding word followed by the candidate $c$.

The first term of the formula is equivalent to the co-occurrence frequency of a long-form candidate with a short form. The second term discounts the first term based on the frequency distribution of nested candidates. Given a long-form candidate $t \in T_c$, $\frac{freq(t)}{\sum_{t \in T_c} freq(t)}$ presents the occurrence probability of candidate $t$ in the nested candidate set $T_c$. Note that $\sum_{t \in T_c} freq(t)$ is not equal to $freq(c)$ only if any contextual sentence beginning with the long form $c$ exists. The second term of the formula calculates the weighted average of the frequency of occurrence of nested candidates accounting for the frequency of candidate $c$. The underlying idea of the subtraction is to disregard the candidate as a part of specific longer candidates. If a long-form candidate $c$ often occurs selectively as a part of a nested candidate $t \in T_c$, $LH(c) \to 0$ as the second term of the formula becomes close to the first term. If

a long-form candidate c does not occur as part of a nested candidate, $LH(c) \rightarrow freq(c)$ as the second term becomes close to zero.

*Comparison with the C-value method*

The original C-value measure (Formula 6.4) subtracts the *non-weighted average* frequency of occurrence of nested candidates $T_c$ (the second term) from the frequency of occurrence of term $w$ (the first term). We have replaced the calculation of average frequency of terms $T_c$ with *weighted average* frequency of terms $T_c$. Just as the Zipf's law [81], a few candidates in $T_c$ account for the most frequency in the total frequency of candidates $T_c$, and the large number of the residual candidates occur rarely. For this reason, if we apply the original definition of the C-value measure to a large collection of text, the effect of the subtraction will be quite diminished, i.e., the second term in Formula 6.4 becomes close to zero because most nested candidates in $T_c$ have low frequency. In contrast, Formula 6.5 subtracts the frequent candidates in $T_c$ from $\texttt{freq}(w)$. In this way, the replacement of the second term with the weighted-average frequency ensures the robustness for a large collection of text with variations and spelling errors.

Let us examine the sample of Figure 31 again. Candidate *factor 1* has four expansions, *transcription factor 1* (dominant one), *nuclear factor 1* (arbitrary variation), *transcription factor 1*, and *transcription factor 1* (spelling mistakes). Even though three out of four expansions are rare/trivial, the original C-value measure deals with all expansions equally. As a result, the original C-value score (without the multiplication of term length) of candidate *factor 1* remains quite high,

$$\text{C-value}(\textit{'factor 1'}) = 216 - \frac{213+1+1+1}{4} = 162. \tag{6.6}$$

The C-value score does not reflect the existing circumstances, implying that the candidate *factor 1* is virtually worth a term appearing 162 times independently of other longer terms. In contrast, the proposed likelihood measure considerably lowers the score of the candidate *factor 1*,

$$\text{LH}(\textit{'factor 1'}) = 216 - \frac{213 \times 213}{216} - \frac{3 \times 1 \times 1}{216} = 5.94. \tag{6.7}$$

Considering that the candidate *factor 1* is highly dependent on the expansion *transcription factor 1*, the score calculated by $LH(c)$ becomes more reasonable.

### 6.3.3 *Extracting authentic long-forms for abbreviations*

Even if the long-form likelihood $LH(c)$ assigns higher scores to a long-form candidate c occurring frequently with a specific abbreviation, this does not assert that the candidate c is the long form for an abbreviation. Table 8 shows a list of long-form candidates for abbreviation *ADM*. The table was generated from 1,314 contextual sentences (containing the abbreviation *ADM* in parentheses) in MEDLINE abstracts. in descending order of their likelihood scores. Candidate *adriamycin* co-occurs the most frequently with abbreviation *ADM*. Since the long-form candidate *adriamycin* contains all letters in the same order as the abbreviation *ADM*, it is considered as an authentic long-form (marked as 'o'). This

*Authentic long-form*

Table 8. Long-form candidates for *ADM*.

| Candidate | Len | Freq | Score | Valid |
|---|---|---|---|---|
| adriamycin | 1 | 727 | 721.4 | 0 |
| adrenomedullin | 1 | 247 | 241.7 | 0 |
| abductor digiti minimi | 3 | 78 | 74.9 | 0 |
| doxorubicin | 1 | 56 | 54.6 | x (missing letters) |
| effect of adriamycin | 3 | 25 | 23.6 | x (expansion) |
| adrenodemedullated | 1 | 19 | 17.7 | 0 |
| acellular dermal matrix | 3 | 17 | 15.9 | 0 |
| peptide adrenomedullin | 2 | 17 | 15.1 | x (expansion) |
| effects of adrenomedullin | 3 | 15 | 13.2 | x (expansion) |
| resistance to adriamycin | 3 | 15 | 13.2 | x (expansion) |
| amyopathic dermatomyositis | 2 | 14 | 12.8 | 0 |
| brevis and abductor digiti minimi | 5 | 11 | 9.8 | x (expansion) |
| minimi | 1 | 83 | 5.8 | x (nested) |
| digiti minimi | 2 | 80 | 3.9 | x (nested) |

is also true for the second and third candidates (*adrenomedullin* and *abductor digiti minimi*).

The fourth candidate *doxorubicin* is interesting, i.e., its score is high although it lacks the necessary letters *a* and *m* for *ADM*. This is because *doxorubicin* is a synonym of *adriamcycin*, and many authors give *ADM* in parentheses following the word without the proper long form (*adriamcycin*). In this case, although the strong co-occurrence between *doxorubicin* and *ADM* implies a meaningful relation, we do not extract such pairs, counting them as invalid (not a proper pair of short/long form).

*Other relations expressed by parentheses*

Most studies (e.g., [1, 73, 77]) introduce a rule to validate a long form for a short form: "all (alphanumeric) letters in a short form must appear in the corresponding long form in the same order." However, one advantage of our approach over the previous work based on letter matching is that it can suggest, based on statistics, a long form whose short form is arranged in a different word order, e.g., *water activity (AW)* and *beta 2 adrenergic receptor (ADRB2)*. Hence, we accept a long-form candidate if the words in the long-form candidate can be rearranged so that all alphanumeric letters in the short form appear in the rearranged long-form candidate in the same order. For example, the long-form candidate *beta 2 adrenergic receptor (ADRB2)* is recognized as a valid expression since the words in the candidate are rearranged as *adrenergic receptor beta 2 (ADRB2)*. In contrast, the long form candidate *rate* for abbreviation *ER* is rejected because the letters 'e' and 'r' appear in the same word so that changing the word order cannot resolve the order discrepancy between the short form and long form.

*Letter-matching*

*Letter-matching with shuffling letters*

To explain the validation algorithm, we define operation *map*: given a letter `s` in a short form, to choose a letter `l` in the corresponding long form which is the same as has not been chosen before.

The validation process can be formalized as a problem of mapping deciding the original positions of letters in a short from as follows:

- From before backward, each alphanumeric letter in a short form must be mapped to the same (case-insensitive) letter in the corre-

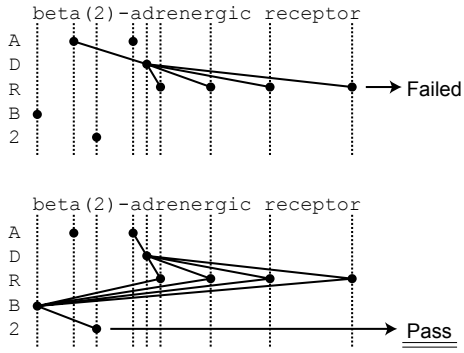



Figure 32. The long-form validation algorithm applied to abbreviation *ADRB2*.

sponding long form.

- After choosing the original location of a letter in a short form, the succeeding letter in the short form must choose the original location letter in the same word or a letter in a mapping a letter in the short form to a letter in a word in the corresponding long form, the succeeding letter in the short form must be mapped to a letter in the same word
- If the algorithm cannot find a position to where the current letter in the short form
- If the algorithm finds a position for the last letter in the short form is mapped to a letter in the long form successfully, the long form is considered as valid.

*Expansion candidate*

We call the fifth candidate *effect of adriamycin* an *expansion* of a long-form since it consists of the authentic long-form *adriamycin* with some preceding words (i.e., *effect of*). As *adriamycin* has a higher score than this candidate, we can disregard the expansion candidates such as *effect of adriamycin* and *resistance to adriamycin* (marked as 'expansion') because they contain unnecessary elements (i.e., *effect of* and *resistance to*) attached to the long form. Similarly, we also disregard nested candidates such as *minimi* and *digiti minimi* (marked as 'nested') since they lack the necessary elements (i.e., *abductor digiti* and *abductor*) to create the correct long-form *abductor digiti minimi*. The likelihood score $\mathrm{LH}(w)$ determines the most appropriate long-form among similar candidates sharing the same words or lacking some words.

*Nested candidate*

We do not include candidates with scores below a given threshold. Therefore, the proposed method cannot extract candidates appearing rarely in the text collection. It depends on the application and considerations of the trade-off between precision and recall, whether or not an abbreviation recognition system should extract such rare long forms. When integrating the proposed method with e.g., Schwartz and Hearst's algorithm, we treat candidates recognized by the external method as if they pass the score cut-off. In Table 8, for example, candidate *automated digital microscopy* is inserted into the result set whereas

```
# [Variables]
#   sf: the short form for which long forms are recognized.
#   candidates: the long-form candidates to be validated.
#   result: the list of authentic long-forms (output).

# Sort long-form candidates in descending order of scores.
candidates.sort(key=lambda lf:lf.score, reverse=True)

# Initialize the result list as empty.
result = []

# Pick up a long form one by one from candidates.
for lf in candidates:
    # Apply a cut-off based on the score...(a)
    if lf.score < 2.0:
        continue
    # Check the letters and their order...(b)
    if not valid_longform(sf, lf):
        continue
    # Apply pruning of redundant long form...(c)
    if redundant(result, lf):
        continue
    # Insert this long form to the result list.
    result.append(lf)

# Output the authentic long-forms.
print result
```

Figure 33. Pseudo-code for extracting authentic long-forms.

candidate *adrenomedullin concentration* is skipped since it is nested by candidate *adrenomedullin*.

Figure 33 contains the pseudo-code for extracting authentic long-forms from the list of long-form candidates and their likelihood scores. Long-form candidates are stored in a list (`candidates`) and are to be validated one by one in descending likelihood order. A long-form candidate is considered valid if the following conditions are met: *(a)* it has a likelihood score $\geqslant 2.0$ (i.e., a long-form candidate must appear at least twice); *(b)* the words in the long form can be rearranged so that all alphanumeric letters in the short form appear in the same order; and *(c)* it is not nested or an expansion of the previously chosen long forms.

*Long-form validation algorithm*

### 6.3.4 *Implementation*

The implemented system first enumerates all short forms in a given text which are likely to be abbreviations by focusing on parenthetical expressions (see Pattern (6.1)). Following the heuristic rules [73], we regard parenthetical expressions as short forms if they consist of at most two words; their length is between two to ten characters; they contain at least an alphabetic letter; and the first character is alphanumeric. All sentences containing a short form are associated with their short forms in a database for efficient access by later processes. For each short form in the database, the system retrieves all contextual sentences for that short form and generates a list of long-form candidates and their likelihood scores. The algorithm described in Section 6.3.3 determines the authentic long-forms in the list. Iterating this process for all short forms, the system yields the list of abbreviations and their expanded forms.

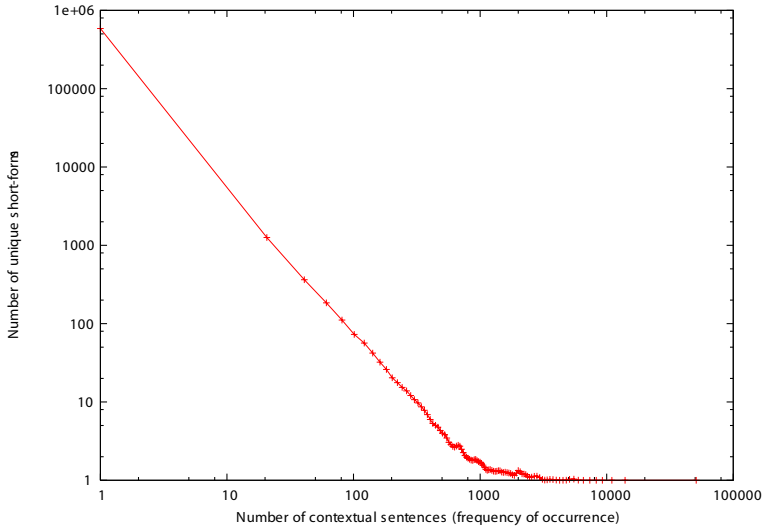



Figure 34. Number of unique short-forms over their frequency of occurrence.

The system is implemented in C/C++ (mainly) and Python (for preprocessing). Contextual sentences are compressed by zlib[10] and stored in a B-tree database implemented by Berkeley DB[11]. The system outputs a list of abbreviations, their expanded forms, and their likelihood scores in a plain text format, which is suitable for developing an abbreviation-dictionary server with an SQL database engine.

Using a desktop computer running on an Intel Pentium 4 3.40GHz processor with 2GB main memory, the author conducted a feasibility experiment, applying the system to the whole MEDLINE database which contained 7,811,582 abstracts (out of 16,069,250 citations)[12]. It took about 12 hours to recognize 886,755 unique short-forms in the abstracts and to insert 9,223,039 contextual sentences into the intermediate database. The short form occurring the most frequently in the abstracts was *II* (50,923 times), followed by *CT* (32,507 times), *III* (30,184 times), *P<0.05* (27,284 times), *PCR* (26,486 times), etc. Some of the candidates such as *III* and *P<0.05* are not real short-forms even though they often appear in parentheses in scientific articles. We do not provide any processing stage in short-form mining to exclude them since they are unlikely to be accompanied by specific long-form candidates and, therefore, to be qualified in the subsequent stages. Some short forms (especially, *III* and *P<0.05*) here are rather common parenthetical expressions in scientific articles than proper abbreviations. Although we can remove such parenthetical expressions by introducing a stop-list for abbreviations, they are also unlikely to have long-form candidates with high likelihood scores, not co-occurring with a specific expression appearing before the parentheses.

Figure 34 shows the relationship between the number of unique short-

10 http://www.zlib.net/

11 http://www.sleepycat.com/products/bdb.html

12 The MEDLINE database was up-to-date on March 2006. The size of the input data amounted to 52GB (from medline06n0001.xml to medline05n0514.xml)

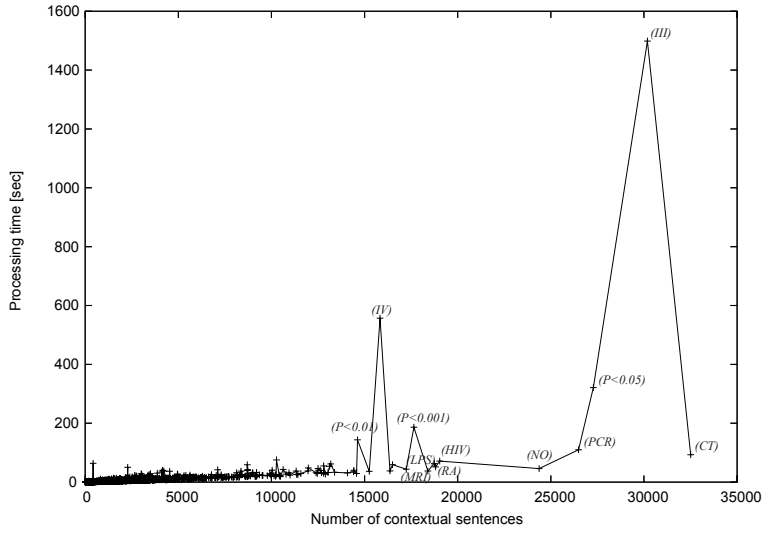



Figure 35. Processing time for different numbers of contextual sentences.

forms and the number of their contextual sentences (i.e., frequency of occurrence of short forms). As Ziph's law [81] suggested, a great deal of short forms identified by the short-form mining occur rarely, but a few kinds of short forms occur frequently in the abstracts. The author continued the subsequent steps of the feasibility experiment with 300,954 unique short-forms appearing in two or more contextual sentences. It took about 35 hours to generate 182,585 unique pairs of short/long forms. Figure 35 shows the processing time for extracting authentic long-forms from different numbers of contextual sentences. For instance, the long forms for *CT* were extracted from 32,507 contextual sentences in 93 seconds, *III* (30,184 sentences) in 1498 seconds, *P<0.05* (27,284 sentences) in 321 seconds, *PCR* (26,486 sentences) in 110 seconds, *NO* (24,369 sentences) in 46 seconds. In general, the processing time depends on the number of long-form candidates generated from the contextual sentences. For this reason, some parenthetical expressions such as *III*, *P<0.05*, *P<0.001*, and *IV* consumed much more time than other short forms due to the diverse expressions appearing before the parentheses. In contrast, the system identified the long forms for the proper abbreviations (e.g., *CT*, *PCR*, and *NO*) in a few minutes. These experimental results reveal that it is feasible to construct an abbreviation dictionary from the whole MEDLINE abstracts with the proposed method.

## 6.4 EXPERIMENTS WITH ENGLISH ABBREVIATIONS

The author compared our method with three baseline systems and two variants of our method.

- **Proposed method (AM)**: described in this paper.

- **Schwartz and Hearst's method (SH)**: Their implementation[13] was used as is.
- **Adar's method (SaRAD)**[14]: The author implemented the abbreviation-recognition component in SaRAD described in the paper [1] and its supplementary information[15]. The implementation is available on our web site.
- **Liu and Friedman's method (LF)**[14]: The author implemented an abbreviation-recognition program described in the paper [39]. The program receives the long-form candidates obtained from the method described in Section 6.3.2 and applies selecting, subsuming, and separating to the long-form candidates. I did not use the SPECIALIST Lexicon (suggested in their paper) for normalizing term-forms, but Porter's stemming algorithm. Having the same set of long-form candidates as a set of potential collocations, we compare the quality of collocation mining with the proposed method. This implementation is also available on our web site.
- **Proposed method with C-value termhood (CV)**: This is a variant of the proposed method applying the C-value measure $CV(c)$ described in Formula 6.4. A comparison between AM and CV will show the improvement of the likelihood measure.
- **Proposed method with Frequency termhood (FREQ)**: This is a variant of the proposed method replacing the likelihood $LH(c)$ with the frequency of occurrence of long-form candidate $c$.

Given a list of target short-forms and their contextual sentences, each system identifies the long forms for the short forms. Porter's stemming algorithm was applied to the long forms in order to match them to reference long-forms extracted by a bio-informatician. We emulate the process of building an abbreviation dictionary by screening long forms that occur $\theta$ or more times in the text collection. In other words, statistical information (frequency of occurrence of long forms) is incorporated even in the letter-matching algorithms as a post-processing phase. For example, setting $\theta$ to 2 implies removing short/long-form pairs occurring once in the text collection, i.e., definitions of dynamic abbreviations. The threshold $\theta$ controls the accuracy/coverage (or precision/recall) tradeoff for an abbreviation dictionary. I drew a precision-recall curve for each system by changing the threshold $\theta$ from 2 to 20.

*Threshold* $\theta$

The author evaluated our method on 637,957 contextual sentences containing 4,024 short/long-form pairs for 100 short-forms[16]: half of the short forms were constituted by the top 50 short forms[17] appearing

13 http://biotext.berkeley.edu/software.html

14 The evaluation results for SaRAD and LF are based on our implementations and might not reflect the actual performance.

15 http://www.hpl.hp.com/research/idl/papers/srad/websup-070703.pdf

16 The 637,957 contextual sentences containing 100 short forms were drawn from the intermediate database described in Section 6.3.4.

17 We have excluded several parenthetical expressions such as *II*, *III*, $P<0.05$, etc. since they do not introduce abbreviations. We have also excluded a few short-forms such as *RA* (18,810 occurrences) and *AD* (17,240 occurrences) because there are too many variations of their expanded forms to handle in manual preparation of our evaluation corpus.

| Rank | Type | Parenthetic phrase | # contextual sentence | # distinct long-forms |
|---|---|---|---|---|
| 1 | c | CT | 32,507 | 257 |
| 2 | c | PCR | 26,486 | 48 |
| 3 | c | HIV | 19,032 | 12 |
| 4 | c | LPS | 18,750 | 52 |
| 5 | c | MRI | 18,396 | 10 |
| .. | c | .... | ... | .. |
| 50 | c | AMI | 5,803 | 47 |
| 51 | t | ATP | 4,993 | 39 |
| 52 | t | PKA | 2,319 | 20 |
| .. | t | .... | ... | .. |
| 75 | t | AW | 376 | 75 |
| 76 | t | TTF-1 | 231 | 1 |
| .. | t | .... | ... | .. |
| 99 | t | CNS1 | 4 | 1 |
| 100 | t | 3-NO2-TYR | 2 | 1 |
| — | | (overall 100 abbreviations) | 637,957 | 4,024 |

Type = { c: top 50 frequent abbreviations, t: appeared in previous papers }

Table 9. Statistics on the evaluation corpus.

most frequently in MEDLINE abstracts; and the remaining 50 short-forms were chosen from those discussed in papers on abbreviation recognition. Note that the use of the frequent 50 short forms for the evaluation *does not* favor our method, which is based on statistics. In fact, a great number of long forms for the 50 short forms were found *rarely*, e.g., as many as 1,076 pairs occur only twice. Although the most frequent long-form for the short form *CT* (32,507 times) is *computed tomography* (18512 times), a great number of less frequent long-forms also exist in the corpus, e.g., *cavernous tissue* (2 times), *complex tone* (2 times), *cortical threshold* (2 times). It is difficult for the proposed method to recognize such rare short/long-form pairs.

*Criteria for including long forms*

When preparing the evaluation collection, criteria for including long forms were established: a long form with minimum necessary elements (words) to produce its abbreviation is accepted; a long form with unnecessary elements, e.g., *magnetic resonance imaging unit (MRI)* or *human immunodeficiency virus infection (HIV)*, is not accepted to keep the criteria for inclusion consistent; a misspelled long-form, e.g., *hidden markvov model (HMM)*, is accepted to separate the abbreviation-recognition task from a spelling-correction task. Expressions satisfying the above criteria were accepted regardless of their popularity or relevance because it is hard for a human subject to determine which long forms are appropriate for the inclusion to a dictionary.

Figure 36 shows the precision-recall curves when we count the number of distinct long-forms, i.e., count once even if short/long form pair ⟨*HMM*, *hidden markov model*⟩ occurs multiple times in the text collection. In general, a system marks the highest recall and lowest precision (i.e., plotted at the left-top in a locus) when the threshold $\theta$ is two. As the threshold $\theta$ increases, the recall and precision become lower and higher respectively (i.e., a locus draws a downward-sloping curve).

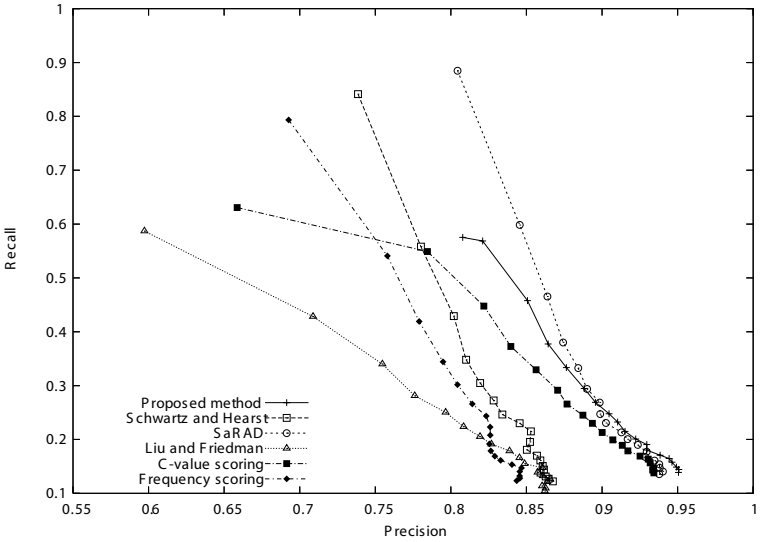



Figure 36. Precision-recall calculated by the distinct numbers of long forms.

The proposed method (AM) achieved 80.8% precision and 57.5% recall at $\theta = 2$, and 95.0% precision and 13.9% recall at $\theta = 20$. When used with a higher threshold ($\theta \geqslant 9$), AM outperformed other methods, marking the highest precision. The simple approach using frequency of co-occurrence (FREQ) did not yield a good result. The comparison between AM and CV also revealed the great improvement of the proposed likelihood over the original C-value measure. These facts strongly suggest the importance of term recognition in statistical long-form recognition.

SaRAD obtained the best result of all systems with a lower threshold, e.g., 80.4% precision and 88.5% recall at $\theta = 2$. This result reflects the advantage of the letter-matching approach when statistical clues in the source text are unavailable. However, SaRAD could not improve the precision so much with a higher threshold. For instance, SaRAD could not utilize frequency information to withdraw a misrecognized long-form *systemic arterial pressure (MAP)* (44 occurrences). AM could grasp that *mean systemic arterial pressure* (42 occurrences) is more appropriate, without complicated tuning of heuristic rules, since 95% of the occurrences of *systemic arterial pressure* are derived from *mean systemic arterial pressure*.

Schwartz and Hearst's method (SH) sufferred from its low precision. This was because SH recognized a number of false long-forms such as *the mean skin temperature (TSK)* (4 occurrences), *a water activity (AW)* (7 occurrences), and *expression of oestrogen receptor (ER)*[18] (18 occurrences). Unlike SaRAD, which is also based on a similar letter matching technique, SH could not remove trailing words that are unnecessary to form an acronym such as *circular dichroism spectroscopy (CD)* (23 occurrences)

18 SH has a constraint that the initial word of a long form must begin with the initial letter of the corresponding short-form. Therefore, SH could not recognize *oestrogen receptor* as a long form although *oestrogen* is an orthographic variant of *estrogen*.

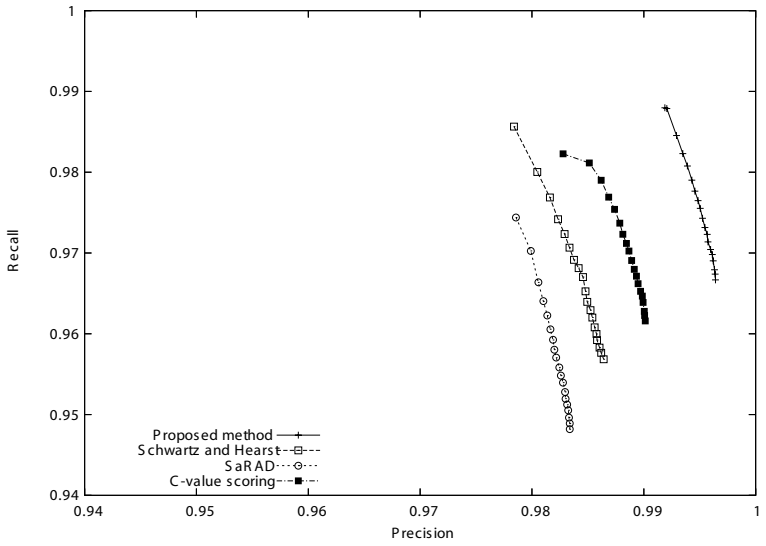



Figure 37. Precision-recall calculated by the numbers of long-form occurrences.

and *radiotherapy alone (RT)* (29 occurrences).

Liu and Friedman's method had difficulty in dealing with diverse expressions in a text collection. For instance, LF has the following rule to withdraw some long-form candidates: "remove a set of candidates $T_w$ formed by adding a prefix word to a candidate $w$ if the number of such candidates $T_w$ is greater than a parameter $t_0$". This rule with $t_o = 3$ applied to the collocation *myocardial infarction* removed the proper long-form *acute myocardial infarction* (5,314 occurrences) for the acronym *AMI* because *myocardial infarction* had 15 possible expansions in the MEDLINE abstracts, e.g., *anterior myocardial infarction* (34 occurrences), *phase myocardial infarction* (13 occurrences), *wall myocardial infarction* (5 occurrences), *inferior myocardial infarction* (4 occurrences), etc. For the same reason, LF missed a number of frequent long-forms such as *epidermal growth factor (EGF)* (10,209 occurrences), *high performance liquid chromatography (HPLC)* (8,723 occurrences), *acquired immunodeficiency syndrome (AIDS)* (6,111 occurrences), etc. This flaw might be improved by tweaking the parameters, but I would like to emphasize that our method achieved the result without a parameter.

Figure 37 shows the precision-recall curves when we count the number of positive/negative instances in the source text, e.g., count 188 true positives if a method identifies the acronym *hidden markov model (HMM)* defined 188 times in the source text. This evaluation metric assesses the appropriateness of dealing with frequent long-forms: a system loses precision/recall with this metric if it misrecognizes/missed a frequent long-form in the text collection. I did not plot the precision-recall locuses for CV (86% precision and 85% recall) and LF (77% precision and 60% recall) to focus on the results of superior systems in the figure.

The proposed method (AM) outperformed the other methods, obtaining the highest precision and recall with all thresholds ($2 \leqslant \theta \leqslant 20$). AM achieved 99.1% precision and 98.7% recall at $\theta = 2$, and 99.6%

precision and 96.6% recall at $\theta = 20$. These figures revealed that the proposed method scarcely missed long forms occurring frequently in the evaluation corpus. Interestingly, Schwartz and Hearst's method (SH) was superior to SaRAD with this metric (Figure 37) even though the order of superiority was reversed with the former metric (Figure 36). In addition, the quality difference between SH and FREQ became more prominent with this metric.

In general, an acronym defined in a number of places is likely to be used without its long form, i.e., as a global acronym. Thus, it is critically important for a recognition method to include frequent acronym definitions in an acronym dictionary. The absences/errors of frequent long-forms in a dictionary may decrease the performance of its application because the application is unaware of the cross-linked relation occurring frequently in a text. Figures 36 and 37 demonstrate that the proposed method has a strong advantage for building a comprehensive dictionary with potential global acronyms.

I now describe the false cases found by the proposed method in order to discuss directions for further improvements. The word *treatment* (22 occurrences) was misrecognized as the long-form for the acronym *RT*. This phenomenon was due to: diverse expressions appear before *treatment* in the text collection, e.g., *receive treatment*, *replace treatment*, *regular treatment*, *radiation treatment*, etc; the diverse expansions could not surpass *treatment*, i.e., Formula 6.5 assigned a higher score to *treatment* than to the diverse expressions; *treatment* contains letters 'r' and 't' in the same order as the acronym; and long-form extraction chose *treatment* and removed all the expansions from the candidate list.

Another type of false case was found where a number of expressions that end with *mean* and contain the letters 's' and 'd' in that order were misrecognized as the long forms for the acronym *SD*, e.g., *years and mean*, *expressed as mean*, *increased mean*, *increased from a mean*, etc. In scientific articles, *mean* followed by parenthetical expression of *SD* is used to describe *mean* and *standard deviation* values at the same time, e.g., "the mean (SD) age of the patients was 49.8 (20.9) years." Although this parenthetical expression does not imply an acronym-definition relation, our long-form extraction algorithm tried to find candidates with letters 's' and 'd' before *mean*. These cases occur rarely in the evaluation corpus as Figure 37 suggested, but are expected to be addressed in future work.

## 6.5 JAPANESE PARENTHETICAL EXPRESSIONS

The proposed method, as presented so far, achieved good results for recognizing abbreviations in English documents. Based on statistical term recognition, the method is expected to have language independence, i.e., applicability to other languages such as Japanese. However, the system's actual application is not that simple: the creation process of Japanese abbreviations is much more complicated than that of English ones. The author examined the use of parenthetical expressions in Japanese newspaper articles and classified them into five categories (Table 10). The first three categories are explained in this thesis because the fourth and fifth categories are irrelevant to abbreviation recognition.

*Classification of Japanese parenthesis usages*

*Japanese acronyms*

The first category *acronym* reduces a full form to a shorter form (*short*

| Type | Example (with English translation) |
|---|---|
| Acronym | 東京大学（東大）<br>University of Tokyo (UoT) |
| Acronym with translation | 夜間離着陸訓練（NLP）<br>Night Landing Practice (NLP)<br>ワールドカップ（W杯）<br>World Cup (WC) |
| Alias | 朝鮮民主主義人民共和国（北朝鮮）<br>Democratic People's Republic of Korea (North Korea) |
| Proprety (reading) | 毅然（きぜん）<br>Kizen (kizen)<br>Wii（ウィー）<br>Wii (pronounced as the pronoun 'we') |
| Property (location) | つくば学園都市（茨城県つくば市）<br>Tsukuba Science City (Tsukuba City, Ibaraki Pref.) |
| Property (affiliation) | 岡崎直観（東大）<br>Naoaki Okazaki (UoT) |
| Property (age) | 岡崎直観（27）<br>Naoaki Okazaki (27) |
| Property (member) | 六カ国協議（米朝中韓日露）<br>Six-Party Talks (USA, DPRK, China, ROK, Japan, Russia) |
| Property (weekday) | 1月1日（月）<br>1st January (Mon) |
| Property (performance) | アストロズ（中地区1位）<br>Astros (champion in National League Central) |
| Complementarity | 真摯に（批判を）受け止めている.<br>I take (the criticism) seriously. |
| Others | … （中略）<br>... (snip) |

Table 10. Japanese parenthetical expressions.

*form*) basically by choosing and removing letters in the full form (*long form*). The characteristic of the acronyms is that a long form is paraphrasable to its short form and vice versa, as long as a reader is aware of the association between the short form and long form. In general, the process of generating English acronyms is easily identified. For example, *University of Tokyo* generates the acronym *UoT* by extracting the head letter of each word. Some acronyms shuffle letters extracted from their expanded forms, e.g. *gamma interferon (IFN-GAMMA)*. The generative process of Japanese acronyms is exactly the same as that of English ones. Figure 38 shows three instances of Japanese acronyms. The first acronym (a) takes two Kanji letters from the head of the two words. The second acronym (b) extracts two Kanji letters from the first and third words. Acronym (c) looks interesting: it takes letters at the *end* of first, second, and fourth words in the expanded forms.

*Japanese acronyms with translation*

To further complicate matters, numerous Japanese acronyms do not share letters between short and long forms because of their foreign origins. In Fig. 39-(a), the expanded form *Yakan Ri-chakuriku Kunren* is a Japanese translation of the English term *night landing practice*, but inherits the English acronym *NLP*. The generation process of the abbreviation *W hai* (b) is much more complicated. The English term *World Cup* is translated to a Katakana Japanese *[wa-rudo kappu]*[19], which

19 Japanese language does not make a distinction between 'l' and 'r', i.e., *world* is represented

| | | |
|---|---|---|
| (a) | 東京大学<br>[Tokyo Daigaku]<br>University of Tokyo | 東大<br>[ToDai] |
| (b) | 国立試験研究機関<br>[Kokuritsu Shiken Kenkyu Kikan]<br>National Institute | 国研<br>[KokuKen] |
| (c) | 首都圏中央連絡自動車道<br>[Shuto-ken Chuo Renraku Jidousha-dou]<br>Metropolitan Inter-City Expressway | 圏央道<br>[Ken-O Dou] |

Figure 38. Japanese acronyms.

is a notation to represent the English pronunciation with Japanese consonants and vowels. The first component of the acronym *W* derives from the first word of the English term, *world*. The second component of the acronym *hai* is a translation of the English word *cup* into the corresponding Kanji word.

*Japanese aliases*

The second category presents generic aliases that are not considered as abbreviations, although a formal name is paraphrasable to its alias and vice versa. For example, *Democratic People's Republic of Korea* is abbreviated as *DPRK*, but the country is also known as *North Korea*. This is true also for their Japanese translations: the formal name of the country is *Cho-sen Minshu Shugi Jinmin Kyo-wakoku* (literal translation of the English name); its alias is *Kita Cho-sen*, which stands for *North Korea*. Even though the formal name does not imply the northern part of Korea, the English and Japanese aliases consist of *Korea* (*Cho-sen*) with the locational modifier *North* (*Kita*).

*Describing additional property values*

The third category *property* has various parenthetical usage. The most common among usage patterns in this category is the parenthetical expression "*... A (B) ...*" is interpreted in the following ways.

- *... A, whose X is B, ...* (e.g., *... A, whose members are B, ...*)
- *... A, which/who is/does X B, ...* (e.g., *... A, which is located in B, ...*)

In these interpretations, implicit element *X* specifies the relation between the elements *A* and *B*; also, the elements *A* and *B* are not paraphrasable. For example, the *reading* usage of a parenthetical expression *A (B)* describes the pronunciation *B* after the expression *A*. In newspaper articles, a game console *Wii* might accompany a parenthetical expression, *(pronounced as the pronoun 'we')*, because the spelling is peculiar for an English word. This usage of parentheses is common to Japanese newspaper articles for annotating pronunciations of difficult Kanji words[20].

*Property 'reading'*

In general, it is difficult for a computer to supplement the element *X* hidden in parenthetical expressions. The parenthetical expression

as *wa-rudo*.

20 Japanese uses more Kanji words than those studied at school during the period of compulsory education.

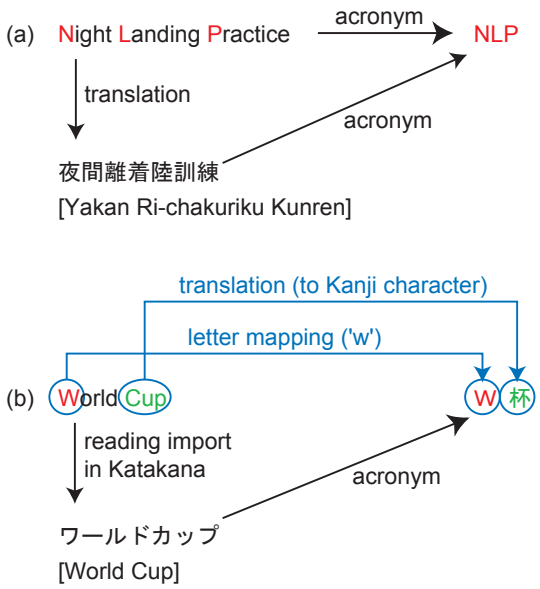



Figure 39. Japanese acronyms with translation.

*Tsukuba Science City (Tsukuba City, Ibaraki Prefecture)* should be interpreted as *Tsukuba Science City, which is located in Tsukuba City, Ibaraki Prefecture* because the inner expression of the parentheses represents a named entity of a location. The parenthetical expression *Naoaki Okazaki (27)* is much more difficult to interpret: the number *27* might represent the age, uniform number, extended telephone number, etc. of the person, depending on the context.

Our goal is to estimate the paraphrase likelihood of two expressions *X* and *Y*, given a parenthetical expression *X (Y)*. From this application point of view, the paraphrase likelihood is expected to discriminate *acronym*, *acronym with translation*, *aliases* from others. In practice, no generic rules or exact patterns have been established for dealing with the various types of lexical paraphrases expressed in parentheses. Consequently, the remainder of this chapter examines an approach to extract parenthetical expressions belonging to the first and second categories, i.e., acronyms (with/without translation) and aliases.

## 6.6 VALIDATING JAPANESE LONG FORMS

*Parenthetical expressions in Japanese newspapers*

We overviewed the complicated generation processes of Japanese abbreviations in the previous section. What would be the result of applying the same method proposed for English documents to Japanese documents? Table 11 illustrates the result with 596,058 articles (8,689,536 sentences) published by the Mainichi Newspapers and the Yomiuri Shimbun in 1998 and 1999.

Similarly to the English abbreviation recognition described in Section 6.3, we collect sentences with parenthetical expressions. We tokenize a sentence with a morphological analyzer to obtain a sequence of words.

| # | Short form | Long form | Score | Para |
|---|---|---|---|---|
| 1 | 北朝鮮<br>(North Korea) | 朝鮮民主主義人民共和国<br>(Democratic People's Republic of Korea) | 4134.1 | o |
| 2 | W杯<br>(W Cup) | ワールドカップ<br>(World Cup) | 2663.9 | o |
| 3 | EU | 欧州連合<br>(European Union) | 2620.8 | o |
| 4 | NATO | 北大西洋条約機構<br>(North Atlantic Treaty Organization) | 2581.9 | o |
| 5 | IMF | 国際通貨基金<br>(International Monetary Fund) | 2461.7 | o |
| 6 | 中国<br>(China) | 人民日報<br>(People's Daily) | 1561.0 | x |
| 7 | IOC | 国際オリンピック委員会<br>(International Olympic Committee) | 1490.5 | o |
| 8 | WTO | 世界貿易機関<br>(World Trade Organization) | 1457.7 | o |
| 9 | 独<br>(Germany) | ディ・ウェルト<br>(Die Welt) | 1386.6 | x |
| 10 | 米<br>(USA) | ワシントン・ポスト<br>(Washington Post) | 1171.0 | x |

Table 11. Top 10 short/long-form pairs in Japanese newspapers.

Extracting long-form candidates that match to the Pattern (6.8)[21],

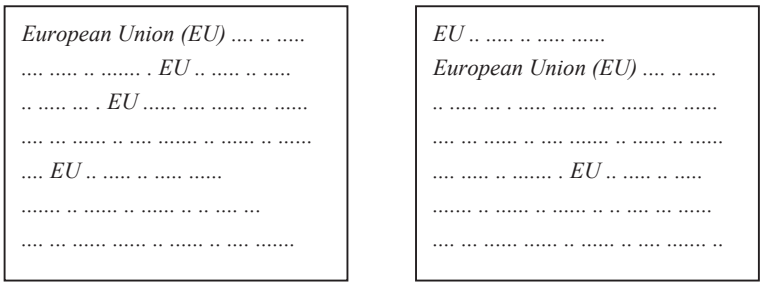
```
.*[:NP:]$                                                    (6.8)
```

we compute the long-form likelihood $\mathrm{LH}(c)$ using the Formula 6.5. Table 11 shows the top 10 short/long-form pairs that have high long-form likelihood scores. The field "Para" presents the paraphrasability of the short/long-form pair: 7 out of 10 pairs in the table are acronyms (#2–5, #7–8) and aliases (#1), but 3 pairs (#6, #9, and #10) express the nationality of the information source.

Table 11 shows clearly that the strong co-occurrence between a long form and a short form does not imply paraphrasability. This study employed a letter-matching algorithm for the English abbreviation recognition to extract 'authentic' long forms. Unfortunately, a naive letter-matching approach might remove all positive cases from this list: 6 out of 7 acronyms entail English translations and therefore do not share letters between short forms and their long forms. Therefore, several validation approaches for Japanese abbreviations are introduced.

STRENGTH OF CO-OCCURRENCE Hisamitsu and Niwa [27] proposed a method for extracting useful parenthetical expressions from Japanese newspaper articles, measuring the co-occurrence strength between the inner and outer phrases of a parenthetical expression. They employed mutual information, $\chi^2$ test with Yates' correction, Dice coefficient, log-likelihood ratio, etc. Although the long-form likelihood proposed in this study also measures the co-occurrence strength, I evaluate the usefulness of other measures, pointwise mutual information $\mathrm{PMI}(A, B)$

21 `[:NP:]` matches a noun phrase, `.*` matches an empty string, noun phrase(s), or any word(s) of any length, and `$` matches a short form of the target abbreviation.




(a) Paraphrase
*g('Europian Union') < g('EU')*

(b) Non paraphrase
*'EU' appears before the parentheses*

Figure 40. Occurrences of paraphrasing.

and $\chi^2(A, B)$:

$$\mathrm{PMI}(A, B) = \log\left[\frac{N \cdot f(A, B)}{f(A)f(B)}\right], \tag{6.9}$$

$$\chi^2(A, B) = \frac{N(ad - bc)^2}{(a + c)(b + d)(a + b)(c + d)}, \tag{6.10}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} f(A, B) & f(A) - f(A, B) \\ f(B) - f(A, B) & N - (a + b + c) \end{pmatrix}. \tag{6.11}$$

In those formulas, $f(A, B)$ denotes the number of sentences containing the expression $A$ followed by the parenthetical expression of $B$; $f(A)$ represents the number of sentences containing the expression $A$; $f(B)$ represents the number of sentences containing the expression $B$; and $N$ is the number of total sentences in the corpus.

LETTER MATCHING As described in Section 6.5, some Japanese long forms restore the letters in their short forms (see acronyms without translation in Table 10). The binary function $\mathrm{MATCH}(A, B)$ is a predicate to check if all letters in the short form $B$ appear in its long form $A$,

$$\mathrm{MATCH}(A, B) = \begin{cases} 1 & \text{(if } A \text{ contains all the letters in } B\text{)} \\ 0 & \text{(otherwise)} \end{cases}. \tag{6.12}$$

This function returns 1 for all cases in Fig. 38 because the long form reproduces the letters in the corresponding short form. In contrast, this function returns 0 for all cases in Fig. 39 because some letters in the short form are missing from the long form.

PARAPHRASE OCCURRENCE It is useful to consider a situation in which a writer introduces the abbreviation or alias $B$ for an expression $A$ in a document. Once the parenthetical expression *A (B)* declares

the paraphrasing $A \rightarrow B$, the writer should prefer the expression $B$ as $A$. The writer should not use the abbreviation or alias $B$ before the parenthetical expression because the expression $B$ might be new to the readers (i.e., this is the writer's motivation to use the parenthetical expression). Therefore, we can estimate the occurrence of paraphrasing by specifically emphasizing the occurrences of the expressions $A$ and $B$ before/after the parenthetical expression.

*Requirements for a paraphrase instance*

Presume that a document has the expression A followed by the parenthetical expression of B. We identify a document as an instance of paraphrasing $A \rightarrow B$ if the document satisfies both of the conditions:

1. The expression B appears more frequently than A after the parenthetical expression *A (B)*.
2. The expression B does not appear before the parenthetical expression.

Figure 40 illustrates two documents with the parenthetical expression *European Union (EU)*. We regard the first document (a) as an instance of paraphrasing, *European Union* $\rightarrow$ *EU* because the expression *EU* occurs more frequently than *European Union* after the parenthetical expression, and because *EU* does not appear before the parenthetical expression. In contrast, document (b) is not considered as a paraphrase instance because *EU* does appear before the parenthetical expression.

*Counting paraphrase occurrences*

Formula 6.13 assesses the probability of paraphrase instances over the parenthetical expressions of $A$ and $B$,

$$\mathrm{PR}(A, B) = \frac{\mathrm{PARA}(A, B)}{\mathrm{D}(A, B)}. \tag{6.13}$$

In this formula, $\mathrm{PARA}(A, B)$ denotes the number of documents satisfying the above conditions; and $\mathrm{D}(A, B)$ presents the number of documents having the expression A followed by the parenthetical expression of B. The function PR(A, B) takes values of 0 (no paraphrase instance) to 1 (all parenthetical expressions introduce the paraphrase) inclusively. The paraphrase occurrence is an original idea used for this study.

SIMILARITY OF LOCAL CONTEXTS Yamamoto [79] proposed the use of similarity of local contexts to measure the paraphrasability of two expressions. His approach is similar to one for Word Sense Disambiguation (WSD) , which exploits the resemblance of the local contexts of the target word and its references. In other words, his assumption is that the sets of words appearing close to the expressions $A$ and $B$ are expected to be similar if the expressions $A$ and $B$ are paraphrasable.

*Word Sense Disambiguation (WSD)*

Following his approach, this study specifically examines each of the words that has a dependency relation from/to the target expression. Collecting such words as the local context of an expression, I use the skew divergence [36] , which is a weighted version of the Kullback-Leibler (KL) divergence , to measure the resemblance of probability distributions P and Q:

*Skew divergence*

*Kullback-Leibler (KL) divergence*

$$\mathrm{SKEW}_\alpha(P||Q) = \mathrm{KL}(P||\alpha Q + (1-\alpha)P), \tag{6.14}$$

$$\mathrm{KL}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{6.15}$$

In these formulas, P is the probability distribution function of the words in the local context for the expression A, Q is that for the expression B, and $\alpha$ is a parameter set to 0.99. The function $\mathrm{SKEW}_\alpha(P\|Q)$ becomes nearly zero if the probability distributions of local contexts for the expressions *A* and *B* are similar.

## 6.7 EXPERIMENTS WITH JAPANESE ABBREVIATIONS

An evaluation was made of abbreviation/alias recognition from 596,058 Japanese articles (8,689,536 sentences) published by the Mainichi Newspapers and the Yomiuri Shimbun in 1998 and 1999. In all, 7,887 short/long-form pairs were obtained from the article collection that satisfy the following conditions:

- the long-form likelihood of a pair is no less than 8;
- a short form does not begin with a numerical letter;
- a short form or long form contains no symbol (e.g., △, ◯, etc.);
- a short form is not registered in the list (see Fig. 41).

The 7,887 short-form and long-form pairs were classified manually into two groups: *paraphrasable* (i.e., acronyms and aliases) and *non-paraphrasable* (i.e., properties, others). The ratio of paraphrasable and non-paraphrasable instances was 1,430 : 6,457.

*Features for SVM classification*

The validation task is also considered as a two-class classification problem. Therefore, I combined the validation approaches presented in the previous section using Support Vector Machines (SVMs). A feature vector for a classification instance (short/long form pair) consists of six values:

- LH: the long-form likelihood;
- PMI: the pointwise mutual information;
- $\chi^2$: the $\chi^2$ value;
- MATCH: the binary value indicating the letter match;
- PR: the probability of paraphrase occurrences; and
- SKEW: the skew divergence of the local contexts.

The parameters for SVMs (e.g., the type of kernel function, the degree in the kernel function) were tweaked slightly, indicating that the fourth-order of polynomial kernel achieved the best accuracy, 89.0% (at ten-fold cross-validation). The precision, recall, and *f*-measure with this optimal parameter were, respectively, 86.5%, 46.6%, and 60.6%. This evaluation result is not comparable to those of previous studies because no previous work assessed results of this validation task. Although

当時 (at that time), 元 (former), 同 (same), 日本時間 (JST)
右 (right), 左 (left), 上 (top), 下 (bottom), 中 (middle), 中央 (middle)
日 (Sun), 月 (Mon), 火 (Tue), 水 (Wed), 木 (Thu), 金 (Fri), 土 (Sat), 祝 (holiday)

Figure 41. Stoplist for Japanese abbreviation recognition.

Table 12. Comparison of different validation approaches

| Method | Accuracy (%) |
|---|---|
| All | 89.0 |
| - LH | 89.0 |
| - MATCH | 89.0 |
| - SKEW | 89.0 |
| - $\chi^2$ | 88.9 |
| - PMI | 88.0 |
| - PR | 82.8 |
| - PR - LH | 82.8 |
| - PR - $\chi^2$ | 82.8 |
| - PR - PMI | 82.6 |
| - PR - SKEW | 82.4 |
| - PR - MATCH | 81.9 |

the abbreviation recognition for English documents achieved much higher accuracy, the proposed method exhibits moderate performance on recognizing Japanese abbreviations/aliases.

*Comparing features*

The author examined the contribution of each feature to the validation task by eliminating features one by one. The concept of this step is that, if a feature is important for classification, then removing the feature will degrade the classification accuracy. Table 12 shows the impact of each feature on the classification task. Each row presents the accuracy of the classification (at ten-fold cross validation) after eliminating some feature(s). For example, the row "- PMI" reports the accuracy of the classification (e.g. 88.0%) after eliminating the PMI feature.

The table reported that the paraphrase-occurrence feature (PR) had the most important influence on the validation task: the absence of this feature reduced the accuracy by 6.2%. Interestingly, the contribution of letter-matching (MATCH), similarity of local contexts (SKEW), and co-occurrence strength (LH, PMI, $\chi^2$) disappeared when PR is available. These facts suggest the importance of the PR feature to estimate the paraphrasability of parenthetical expressions. The letter-matching feature (MATCH) or similarity of local contexts (SKEW) would play an influential rule if the paraphrase-occurrence feature (PR) were unavailable.

## 6.8 SUMMARY

This chapter described a term recognition approach to extract abbreviations and their definitions from a large text collection. The main contribution of this study has been to show the usefulness of statistical information for building an abbreviation dictionary of good quality. The proposed method outperformed the base-line systems, achieving 99% precision and 82–95% recall on our evaluation corpus, which roughly emulates all of MEDLINE. Figures 36 and 37 illustrate the superiority of the proposed method in building a precise and comprehensive abbreviation dictionary. A future direction of this study would be to combine a letter-matching algorithm to improve the recall of recognizing rare short/long-form pairs (if rare pairs are necessary) and to incorporate other types of relations that are expressed with parentheses such as synonyms and paraphrases.

Even though the statistical approach is independent from the target language, the creation process of Japanese abbreviations was much more complicated than that of English ones. Strong co-occurrence does not imply the paraphrasability of a long form to its short form and vice versa. For that reason, this study proposed a method to classify parenthetical expressions into paraphrasable and non-paraphrasable groups. The proposed method achieved a 60.6% *f*-measure on an evaluation corpus constructed from Japanese articles published in Mainichi Newspapers and the Yomiuri Shimbun in 1998 and 1999. A future direction of this study would be, for example, to incorporate a translation model for dealing with abbreviations originating from foreign words.

# Part IV

# CONCLUSION

# 7 CONCLUSION

This thesis examined methodologies for aggregating information and knowledge across documents. As the first step to aggregate information in multiple documents, Chapter 4 described sentence extraction. The problem of sentence extraction was formalized as a combinational optimization problem that determines a set of sentences containing as many important information fragments as possible. The presented system achieved a good result using the TSC-3 evaluation corpus. A comparison among sentence representations demonstrated that the pairwise dependency relation performed better than either bag-of-words or co-occurrence representations.

The sentence extraction presented in Chapter 4 rejected the inclusion of redundant information in a summary. However, the approach did not consider the coherent aggregation of information. Chapter 5 examined a method to arrange the extracted sentences. The main contribution of the work was: to propose a method for the effective use of precedent information for improving chronological ordering; to obtain the optimal combination of four existing criteria for arranging sentences; and to establish the methodology for evaluating sentence ordering, i.e., sentence continuity metric, average continuity metric, and amendment-based evaluation task. These findings will shed light on the problem of modeling coherence in information aggregation, which has been a difficult question in NLP.

Although redundancy similar to that modeled in Chapter 4 was the target of elimination, the abbreviation recognition presented in Chapter 6 leveraged the redundancy of abbreviation definitions across documents. This is another form of information aggregation, i.e., knowledge extraction in source documents for building a useful resource (dictionary). The main contribution of this work was to illustrate the effectiveness of statistical information for building an abbreviation dictionary of good quality.

In conclusion, this thesis presents examination of novel methods for sentence extraction, sentence ordering, and acronym recognition. Even though this thesis targets these topics as exemplary problems, the outcomes of this study will contribute to various NLP applications.

Part V

# APPENDIX

# BIBLIOGRAPHY


[1] Eytan Adar. SaRAD: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533, 2004. (Cited on pages 68, 69, 75, and 80.)

[2] Hiroko Ao and Toshihisa Takagi. ALICE: An algorithm to extract abbreviations from MEDLINE. *Journal of the American Medical Informatics Association*, 12(5):576–586, 2005. (Cited on page 69.)

[3] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA, 1998. Association for Computational Linguistics. (Cited on page 29.)

[4] Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. Headline generation based on statistical translation. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325, Morristown, NJ, USA, 2000. Association for Computational Linguistics. (Cited on page 25.)

[5] Regina Barzilay and Michelle Elhadad. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97) (at ACL'97/EACL'97 Joint Conference)*, pages 10–17, 1997. (Cited on page 28.)

[6] Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference*, pages 113–120, 2004. (Cited on pages 41, 42, and 44.)

[7] Regina Barzilay, Michael Elhadad, and Kathleen R. McKeown. Inferring strategies for sentence ordering in multidocument summarization. *Journal of Artifical Intelligence Research (JAIR)*, 17:35–55, 2002. (Cited on pages 39, 40, 42, 48, 49, 50, 57, and 63.)

[8] Phyllis B. Baxendale. Man-made index for technical literature — an experiment. *IBM Journal of Research and Development*, 2(4): 354–361, 1958. (Cited on page 27.)

[9] Adam Berger and Vibhu O. Mittal. Query-relevant summarization using faqs. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 294–301, Morristown, NJ, USA, 2000. Association for Computational Linguistics. (Cited on page 25.)

[10] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996. (Cited on page 70.)

[11] Harold Borko and Charles Bernier. *Abstracting Concepts and Methods*. Academic Press, San Diego, California, 1975. (Cited on page 10.)

[12] Ronald Brandow, Karl Mitze, and Lisa F. Rau. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31(5):675–685, 1995. (Cited on page 27.)

[13] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990. (Cited on page 70.)

[14] Jeffrey T. Chang and Hinrich Schütze. Abbreviations in biomedical text. In S. Ananiadou and J. McNaught, editors, *Text Mining for Biology and Biomedicine*, pages 99–119. Artech House, Inc., 2006. (Cited on pages 67 and 69.)

[15] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967. (Cited on page 49.)

[16] Edward T. Cremmins. *The Art of Abstracting*. Information Resource Press, Arlington, Virginia, 1996. (Cited on page 11.)

[17] Hal Daumé, III and Daniel Marcu. A noisy-channel model for document compression. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 449–456, Morristown, NJ, USA, 2002. Association for Computational Linguistics. (Cited on pages 25 and 70.)

[18] Harold P. Edmundson. New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 16(2):264–285, 1969. (Cited on pages 27 and 28.)

[19] Brigitte Endres-Niggemeyer. *Summarizing Information*. Springer, Berlin, 1998. (Cited on pages 12 and 25.)

[20] Charles J. Fillmore. The case for case. In Emmon Bach and Robert T. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, RineHart, and Winston, New York, NY, USA, 1968. (Cited on page 29.)

[21] Katerina T. Frantzi and Sophia Ananiadou. The C-value / NC-value domain independent method for multi-word term extraction. *Journal of Natural Language Processing*, 6(3):145–179, 1999. (Cited on pages 72 and 73.)

[22] Sylvain Gaudan, Harald Kirsch, and Dietrich Rebholz-Schuhmann. Resolving abbreviations to their senses in medline. *Bioinformatics*, 21(18):3658–3664, 2005. (Cited on page 68.)

[23] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop on Automatic Summarization*, volume 4, pages 40–48, 2000. (Cited on page 29.)

[24] Tsutomu Hirao, Hideki Isozaki, Eisaku Maeda, and Yuji Matsumoto. Important sentence extraction based on support vector machines. In *Proceedings of COLING-2002*, pages 342–348, 2002. (Cited on page 28.)

[25] Tsutomu Hirao, Manabu Okumura, Takahiro Fukushima, and Hidetsugu Nanba. Text Summarization Challenge 3 — Text summarization evaluation at ntcir workshop 4 —. In *NTCIR Workshop 4: Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization*, 2004. (Cited on pages 17 and 34.)

[26] Tsutomu Hirao, Manabu Okumura, Takahiro Fukushima, Hidetsugu Nanba, and Chikashi Nobata. Corpus and evaluation measures for multiple document summarization with multiple sources. In *Proc. of the 20th International Conference on Computational Linguistics (COLING2004)*, volume 1, pages 535–541, 2004. (Cited on page 18.)

[27] Toru Hisamitsu and Yoshiki Niwa. Extracting useful terms from parenthetical expression by combining simple rules and statistical measures: A comparative evaluation of bigram statistics. In Didier Bourigault, Christian Jacquemin, and Marie-C L'Homme, editors, *Recent Advances in Computational Terminology*, pages 209–224. John Benjamins, 2001. (Cited on pages 69, 70, and 88.)

[28] Jerry R. Hobbs. *Literature and Cognition*. Center for the Study of Language and Information (CSLI). The University of Chicago Press, 1990. (Cited on page 40.)

[29] Eduard Hovy. Automated text summarization. In Ruslan Mitkov, editor, *The Oxford Handbook of Computational Linguistics*. Oxford University Press, Oxford, UK, 2001. (Cited on page 9.)

[30] David Hume. *An Enquiry Concerning Human Understanding*. Project Gutenberg (Online Book Catalog), 1748. (Cited on page 40.)

[31] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Document retrieval systems*, pages 132–142, 1988. (Cited on page 27.)

[32] Donald E. Knuth. Computer programming as an art. *Commun. ACM*, 17(12):667–673, 1974. (Cited on page xi.)

[33] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops)*, pages 63–69, 2002. (Cited on pages 20 and 29.)

[34] Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73, New York, NY, USA, 1995. ACM Press. (Cited on page 28.)

[35] Mirella Lapata. Probabilistic text structuring: experiments with sentence ordering. In *Proceedings of the 41st Meeting of the Association of Computational Linguistics*, pages 545–552, 2003. (Cited on pages 41, 42, 44, 61, and 63.)

[36] Lillian Lee. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001*, pages 65–72, 2001. (Cited on page 90.)

[37] Chin-Yew Lin and Eduard Hovy. NEATS: A multidocument summarizer. In *Proceedings of the Document Understanding Conference (DUC01)*, Aug. 2001. (Cited on pages 40, 42, 46, and 57.)

[38] Chin-Yew Lin and Eduard Hovy. From single to multi-document summarization: a prototype system and its evaluation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 457–464, Morristown, NJ, USA, 2002. Association for Computational Linguistics. (Cited on pages 29 and 46.)

[39] Hongfang Liu and Carol Friedman. Mining terminological knowledge in large biomedical corpora. In *8th Pacific Symposium on Biocomputing (PSB 2003)*, pages 415–426, 2003. (Cited on pages 69, 70, and 80.)

[40] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM journal of Research and Development*, 2(2):159–165, 1958. (Cited on pages 26, 27, and 39.)

[41] Inderjeet Mani. *Automatic Summarization*. John Benjamins, Amsterdam/Philadelphia, 2001. (Cited on pages 4 and 9.)

[42] Inderjeet Mani and Eric Bloedorn. Multidocument summarization by graph search and matching. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 622–628, 1997. (Cited on page 28.)

[43] Inderjeet Mani and Eric Bloedorn. Machine learning of generic and user-focused summarization. In *The 15th National Conference on Artificial Intelligence*, pages 821–826, 1998. (Cited on page 28.)

[44] Inderjeet Mani and George Wilson. Robust temporal processing of news. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 69–76, Morristown, NJ, USA, 2000. Association for Computational Linguistics. (Cited on page 47.)

[45] Inderjeet Mani, Barry Schiffman, and Jianping Zhang. Inferring temporal ordering of events in news. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 55–57, Morristown, NJ, USA, 2003. Association for Computational Linguistics. (Cited on page 47.)

[46] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8: 243–281, 1988. (Cited on page 40.)

[47] Daniel Marcu. From local to global coherence: A bottom-up approach to text planning. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 629–635, Providence, Rhode Island, 1997. (Cited on page 55.)

[48] Daniel Marcu. The rhetorical parsing of natural language texts. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 96–103, Morristown, NJ, USA, 1997. Association for Computational Linguistics. (Cited on page 28.)

[49] Daniel Marcu. Discourse trees are good indicators of importance in text. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*. MIT Press, 1999. (Cited on page 28.)

[50] Kathleen R. McKeown, Judith L. Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. Towards multidocument summarization by reformulation: progress and prospects. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 453–460, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence. (Cited on pages 40, 42, and 57.)

[51] Norifumi Murayama and Manabu Okumura. Noisy-channel model wo mochita ryakugo jidou suitei (automatic estimation of abbreviaion using noisy-channel model). In *Proceeding of Gengo Shori Gakkai Dai 12 Kai Nenji Taikai (Proceeding of the 12th Japanese Domestic Conference on Natural Language Processing)*, pages 763–766, 2006. (Cited on page 70.)

[52] David Nadeau and Peter D. Turney. A supervised learning approach to acronym identification. In *8th Canadian Conference on Artificial Intelligence (AI'2005) (LNAI 3501)*, page 10 pages, 2005. (Cited on page 69.)

[53] Katashi Nagao and Koichi Hasida. Automatic text summarization based on the global document annotation. In *Proceedings of the 17th International Conference on Computational Linguistics / 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL '98)*, pages 917–921, Montreal, Quebec, Canada, Aug. 1998. (Cited on page 29.)

[54] Naoaki Okazaki, Yutaka Matsuo, Naohiro Matsumura, Hironori Tomobe, and Mitsuru Ishizuka. Two different methods at NTCIR3-TSC2: Coverage oriented and focus oriented. In *Working Notes of the Third NTCIR Workshop Meeting, Part V: Text Summarization Challenge 2 (TSC2)*, pages 39–46, 2002. (Cited on page 29.)

[55] Naoaki Okazaki, Yutaka Matsuo, Naohiro Matsumura, and Mitsuru Ishizuka. Sentence extraction by spreading activation with similarity measure. *IEICE Transactions on Information and Systems (Special Issue on Text Processing for Information Access)*, E86-D(9): 915–926, 2003. (Cited on page 28.)

[56] Manabu Okumura and Hidetsugu Nanba. Automated text summarization: A survey (in japanese). *Shizen Gengo Shori (Japanese Journal of Natural Language Processing)*, 6(6):1–26, 1999. (Cited on page 26.)

[57] Chris D. Paice. The automatic generation of literature abstracts: an approach based on the identification of self-indicating phrases. In *SIGIR '80: Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, pages 172–191, Kent, UK, 1981. Butterworth & Co. (Cited on page 27.)

[58] Chris D. Paice. Constructing literature abstracts by computer: techniques and prospects. *Information Processing and Management*, 26(1):171–186, 1990. (Cited on page 26.)

[59] Serguei Pakhomov. Semi-supervised maximum entropy based approach to acronym and abbreviation normalization in medical texts. In *40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, 2002. (Cited on page 69.)

[60] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, 2002. (Cited on page 46.)

[61] Youngja Park and Roy J. Byrd. Hybrid text mining for finding abbreviations and their definitions. In *2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 126–133, 2001. (Cited on page 73.)

[62] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3): 130–137, 1980. (Cited on page 72.)

[63] James Pustejovsky, José Castaño, Brent Cochran, Maciej Kotecki, and Michael Morrell. Automatic extraction of acronym meaning pairs from MEDLINE databases. *MEDINFO 2001*, pages 371–375, 2001. (Cited on page 67.)

[64] Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. MEAD - a platform for multidocument multilingual text summarization. In *4th International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal, May 2004. (Cited on page 15.)

[65] Dragomir Radev, Jahna Otterbacher, Adam Winkel, and Sasha Blair-Goldensohn. Newsinessence: summarizing online news topics. *Communications of the ACM*, 48(10):95–98, 2005. (Cited on page 14.)

[66] Dragomir R. Radev and K. McKeown. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500, 1998. (Cited on page 5.)

[67] Dragomir R. Radev, H. Jing, and M. Budzikowska. Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies. In *The ANLP/NAACL2000*

*Workshop on Automatic Summarization*, pages 21–30, 2000. (Cited on page 29.)

[68] Hiroyuki Sakai and Shigeru Masuyama. Improvement of the method for acquiring knowledge from a single corpus on correspondences between abbreviations and their original words. *Shizen Gengo Shori (Japanese Journal of Natural Language Processing)*, 12(4), 2005. (Cited on page 69.)

[69] Gerard Salton and Chung-Shu Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, 29(4):351–372, 1973. (Cited on page 27.)

[70] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11): 613–620, 1975. (Cited on pages 29 and 49.)

[71] Gerard Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. *Information Processing and Management*, 32(2):53–65, 1997. (Cited on page 28.)

[72] Roger C. Schank. *Conceptual Information Processing*. Elsevier Science Inc., New York, NY, USA, 1975. (Cited on page 29.)

[73] Ariel S. Schwartz and Marti A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing (PSB 2003)*, number 8, pages 451–462, 2003. (Cited on pages 69, 75, and 77.)

[74] Karen Sparck-Jones. Automatic summarizing: Factors and directions. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*. MIT Press, 1999. (Cited on page 9.)

[75] Kazem Taghva and Jeff Gilbreth. Recognizing acronyms and their definitions. *International Journal on Document Analysis and Recognition (IJDAR)*, 1(4):191–198, 1999. (Cited on page 69.)

[76] Yoshimasa Tsuruoka, Sophia Ananiadou, and Jun'ichi Tsujii. A machine learning approach to acronym generation. In *Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, pages 25–31, Detroit, USA, June 2005. Association for Computational Linguistics. (Cited on page 70.)

[77] Jonathan D. Wren and Harold R. Garner. Heuristics for identification of acronym-definition patterns within text: towards an automated construction of comprehensive acronym-definition dictionaries. *Methods of Information in Medicine*, 41(5):426–434, 2002. (Cited on pages 69 and 75.)

[78] Jonathan D. Wren, Jeffrey T. Chang, James Pustejovsky, Eytan Adar, Harold R. Garner, and Russ B. Altman. Biomedical term mapping databases. *Database Issue*, 33:D289–D293, 2005. (Cited on page 67.)

[79] Kazuhide Yamamoto. Acquisition of lexical paraphrases from texts. In *2nd International Workshop on Computational Terminology (Computerm 2002, in conjunction with Coling2002)*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics. (Cited on page 90.)

[80] Hong Yu, George Hripcsak, and Carol Friedman. Mapping abbreviations to full forms in biomedical articles. *Journal of the American Medical Informatics Association*, 9(3):262–272, 2002. (Cited on page 69.)

[81] George K. Zipf. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge MA, 1949. (Cited on pages 74 and 79.)

## COLOPHON

This thesis was typeset with LaTeX $2_\varepsilon$ with André Miede's *A Classic Thesis Style* version 1.3.1 (August 2006). The style uses Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL* were used). The listings are typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.) The typographic style was inspired by Robert Bringhurst's genius as presented in *The Elements of Typographic Style* (Version 2.5, Hartley & Marks, 2002).

*Final Version* as of January 28, 2007 at 16:47.

# DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

*7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, JAPAN,*

Naoaki Okazaki