AUTOMATED BIOMEDICAL TEXT FRAGMENTATION

IN SUPPORT OF
BIOMEDICAL SENTENCE FRAGMENT CLASSIFICATION

by

SARA SALEHI

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen’s University
Kingston, Ontario, Canada
September 2009

Copyright © Sara Salehi, 2009
Abstract

The past decade has seen a tremendous growth in the amount of biomedical literature, specifically in the area of bioinformatics. As a result, biomedical text categorization has become a central task for providing researchers with literature appropriate for their specific information needs.

Pan et al. [33, 49] have explored a method that automatically identifies information-bearing sentence fragments within scientific text. Their proposed method aims to automatically classify sentence fragments into certain sets of categories defined to satisfy specific types of information needs. The categories are grouped into five different dimensions known as Focus, Polarity, Certainty, Evidence, and Trend. The reason that fragments are used as the unit of classification is that the class value along each of these dimensions can change mid-sentence.

In order to automatically annotate sentence fragments along the five dimensions, automatically breaking sentences into fragments is a necessary step. The performance of the classifier depends on the sentence fragments. In this study, we investigate the problem of automatic fragmentation of biomedical sentences, which is a fundamental layer in the multi-dimensional fragment classification. In addition, we believe that our proposed fragmentation algorithm can be used in other domains such as sentiment analysis. The goal of sentiment analysis is often to classify the polarity (positive or
negative) of a given text. Sentiment classification can be conducted at different levels such as document, sentence, or phrase (fragment) level. Our proposed fragmentation algorithm can be used as a prerequisite for phrase-level sentiment categorization which aims to automatically capture multiple sentiments within a sentence.
Acknowledgments

First, I would like to thank my supervisor Dr. Hagit Shatkay for her guidance and expertise throughout the course of this thesis work. I have benefitted immensely from her high academic standards and her dedication to research.

I would like to extend my gratitude to my thesis examination committee, Dr. Randy E. Ellis, Dr. Greg Lessard, and Dr. Janice Glasgow for their insightful suggestions.

I would also like to extend a special thanks to my friends Phil Hyoun Lee, Hassan Sayyadi, Amir Tahmasebi, and Anton Oentoro for all of their valuable comments.

I extend my thanks to Queen’s University for the financial support I received while completing my degree.

At last, but far from least, I would like to deeply thank my wonderful parents, Mahnaz and Hamid, and my lovely sister, Mona, for their loving support and endless encouragement. I could not have completed this research without your unconditional support - I love you lots.
## Contents

Abstract .......................................................... i
Acknowledgments ................................................... iii
Contents ........................................................... iv
List of Tables ...................................................... vii
List of Figures ..................................................... viii

1 Introduction ...................................................... 1
   1.1 Motivation ..................................................... 2
   1.2 Thesis Objectives ............................................. 3
   1.3 Thesis Contributions ......................................... 4
   1.4 Thesis Organization ......................................... 4

2 Background and Related Work ................................ 6
   2.1 Introduction to the Text Fragmentation Task ............... 6
   2.2 Fragment Annotation ......................................... 7
   2.3 Fragment Classification ...................................... 9
      2.3.1 Text Preprocessing ...................................... 10
      2.3.2 Maximum Entropy Model for Fragment Classification ... 13
   2.4 Entropy and Certainty ....................................... 15
      2.4.1 The Maximum and Minimum values of the Entropy ....... 17
      2.4.2 Joint Entropy ............................................ 18
   2.5 Related Work ................................................ 18
      2.5.1 Sentence Segmentation .................................. 19
      2.5.2 Intra-Sentence Segmentation ............................ 20
      2.5.3 Topic Segmentation ...................................... 21
      2.5.4 Related Applications ................................... 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1.5</td>
<td>To randomly fragment sentences</td>
<td>98</td>
</tr>
<tr>
<td>B.1.6</td>
<td>To calculate the edit-distance between ground truth fragmentations and the fragmentations resulting from our algorithm</td>
<td>98</td>
</tr>
<tr>
<td>B.1.7</td>
<td>To calculate the edit-distance between ground truth fragmentations and the randomly generated fragmentations</td>
<td>99</td>
</tr>
</tbody>
</table>
## List of Tables

2.1 Selecting terms along the *Focus* and *Evidence* dimensions based on their part-of-speech tags. F denotes *Focus* and E denotes *Evidence*. Terms with PoS tags marked as 1 are selected while those marked with 0 are not selected. ................................. 12

4.1 **The match-move procedure** ............................... 46

5.1 *Dataset information*: Sent-FE generated based on the annotation agreement of all three annotators. Sent-FE-INFO contains only informative sentences in Sent-FE. .................................................... 52

5.2 *The performance of the Focus classification for 1804 sentences (1844 fragments)*. S denotes Scientific, G denotes Generic, and M denotes Methodology. ................................................................. 72

5.3 *The performance of the Evidence classification for 1804 sentences (1844 fragments)*. E0 denotes No evidence, E1 denotes Claim of evidence without verifying information, and E2 denotes Explicit citation, and E3 denotes Explicit evidence. ............................................................. 72

5.4 *The confusion matrix of the Focus classification for 80 fragments*. Each row represents a true class label, while each column represents a predicted class label. ............................................................. 73

5.5 *The confusion matrix of the Evidence classification for 80 fragments*. 73

5.6 *The confusion matrix of the Focus classification for 470 fragments*. 74

5.7 *The confusion matrix of the Evidence classification for 470 fragments*. 74
List of Figures

3.1 Possible fragmentations for a sentence consisting of 4 tokens. Class assignments are shown for the fragments of fragmentation 2 and 8. 

4.1 Ref is the reference fragmentation, Frag(S), while Prop is the proposed fragmentation, Frag'(S). Tokens and separators are shown in the two fragmentations.

4.2 The proposed separator at position 4 must be matched to the reference separator at position 3. Moreover, the proposed separator at position 6 must be matched to the reference separator at position 7.

5.1 The distribution of the number of fragments for the human-annotated fragmentations.

5.2 The distribution of the index of the fragmentation point normalized by the length of a sentence, for 40 sentences that were also fragmented by the human annotators.

5.3 Histogram of number of tokens for 132 human-annotated fragments. This diagram uses the Quantity column from the table shown on the top right-hand side. The first column in the table represents the left point side of each interval. The second column is the interval width, and the third column (Quantity) represents the number of sentences that fall into each interval.

5.4 Histogram of number of informative fragmentations for all sentences. Table on the right corner lists the quantities being used in the histogram and consists of three columns. The first column represents the left point side of each interval. The second column is the interval width, and the third column represents the number of sentences that fall into each interval.

5.5 The distribution of joint entropy values for the highest scoring fragmentations.

5.6 The distribution of number of fragments for sentences fragmented by the fragmentation algorithm.
5.7 Edit-distance values for 1804 sentences. ........................................... 65
5.8 Edit-distance values for the 40 sentences that were also fragmented by
human annotators. ................................................................. 66
5.9 Edit-distance values between randomly generated fragmentations and
ground truth for 1804 fragmentations. ........................................ 67
5.10 Edit-distance values between randomly generated fragmentations and
ground truth for 40 fragmentations that were also fragmented by hu-
man annotators. The grey part represents the case when the algorithm
randomly generating fragmentations, leaves the sentences unfragmented
while the human annotators fragmented the sentences into two fragments. 68
Chapter 1

Introduction

A tremendous growth in the amount of biomedical literature has made it challenging for researchers to find information most relevant and useful for specific analysis tasks. As a result, categorization of biomedical literature has become a central task for providing researchers with literature appropriate for their specific information needs.

We note that different researchers have different needs and expectations for searching. For instance, database curators are often interested in sentences rich in experimental evidence and methodological details, while for a researcher looking for information about a certain gene, the sentences mentioning this gene are of interest. By first identifying text segments rich in the type of scientific content that is of interest to the user, text mining applications can cater to specific users needs.

To determine whether the text is useful to a particular user with specific information needs, Wilbur et al. [53] introduced a set of categories defined to characterize statements made in the literature. The categories are grouped into five dimensions. These dimensions are defined as: **Focus** (*Scientific finding, General knowledge, or Methodology*), **Polarity** (*Positive, or Negative*), **Certainty** (*Complete uncertainty, Low*
certainty, High likelihood, and Complete certainty), Evidence (No evidence, Implication of evidence, Explicit citation, and Explicit evidence), and Trend (Increase or Decrease). The class value along each of these dimensions can change mid-sentence; thus, sentence fragments are used as the unit of categorization.

Pan et al. [33, 49] have explored a text categorization method that automatically characterizes sentence fragments along the five dimensions. The sentence fragments used in their work were all manually fragmented by human annotators. These sentence fragments were used to train their proposed text classifiers. Finally, the proposed text classifiers were used to classify sentence fragments along the five dimensions. Their system can provide curators with candidate sentence fragments that address certain types of information needs.

Our goal in this thesis work is to automate the process of breaking sentences into fragments. The multi-dimensional classifier can then be used to tag each fragment according to the predefined criteria.

1.1 Motivation

As noted above, the multi-dimensional classification method proposed by Pan et al. [33, 49] treats sentence fragments as the basic classification units. Their proposed classifier automatically classifies sentence fragments along the predefined five dimensions. Thus, in order to use automatic classification, one must have automatic fragmentation as well. Additionally, the performance of the multi-dimensional classifier depends on the sentence fragments.

The goal of this thesis is to investigate the problem of automatic fragmentation of biomedical sentences, which is a fundamental layer in the multi-dimensional fragment
classification. The fragmentation is performed using machine learning methods in a way that aims to lead to the optimal classification results in the multi-dimensional fragment classification.

1.2 Thesis Objectives

The work described here aims to investigate the utility of using machine learning methods to break biomedical sentences into their fragments. Our primary goal is to design and develop a fragmentation system that can automatically break biomedical sentences into their fragments. The generated sentence fragments can further be categorized into predefined categories using the classifier proposed by Pan et al. [33, 49]. The core of our approach is based on information theory and minimum entropy. Its development involves the followings essential steps:

1. Generating all the possible fragmentations that meet certain criteria further discussed in Chapter 3, within sentences.

2. Defining a score function using entropy and assigning a score to each of the generated fragmentations within sentences.

3. Selecting the highest scoring fragmentations as optimal within sentences.

We also propose a new evaluation measure that compares fragmentations resulting from applying our proposed algorithm with fragmentations assigned by human annotators. Our proposed automated fragmentation algorithm is described in Chapter 3, followed by a description of our proposed evaluation measure in Chapter 4.
1.3 Thesis Contributions

This thesis proposes a fragmentation algorithm and describes the development of the proposed algorithm used to break biomedical sentences into fragments. We conduct an in-depth study of specific components of our automatic fragmentation algorithm: fragmentation generation and fragmentation scoring. Moreover, we note that unlike human annotators, our fragmentation algorithm may break sentences in the middle of a phrase which leads to generate fragmentations that may be different from those generated by human annotators. Considering our target application, we show that these fragmentations are not necessary incorrect since the fragments within them capture the same information as human annotators fragments.

Furthermore, we discuss standard evaluation measures such as: Precision, Recall, and the traditional edit-distance with only two operations insert and delete, and some of their shortcomings for evaluating the performance of our fragmentation algorithm. We then propose a new evaluation measure, namely, edit-distance with separator-move operation which addresses the limitations related to these standard evaluation measures.

1.4 Thesis Organization

The rest of the thesis is organized as follows: In Chapter 2, we first review the topics of fragment annotation and fragment classification, which are components used in our proposed algorithm. We then provide a broad survey of previous work on the fragmentation task. In Chapter 3, we investigate a novel approach, namely, automated biomedical text fragmentation, to automatically break sentences into fragments. In
Chapter 4, we propose a performance measure that can be applied to general fragmentation tasks. Experimental results and analysis are provided in Chapter 5, followed by conclusions and future work in Chapter 6.
Chapter 2

Background and Related Work

In this chapter, we first define text fragmentation. We then describe fragment annotation, fragment classification, and entropy, which are the major components of the methodology used in this thesis. Finally, we survey previous research on fragmentation.

2.1 Introduction to the Text Fragmentation Task

In natural language processing (NLP), text segmentation is the process of dividing written text into meaningful segments, such as tokens [54], phrases [56], sentences [1, 32], or topics [15, 42, 2, 13].

Text segmentation has a wide range of applications such as information retrieval and text summarization. Depending on the application, segments may be tokens, phrases, sentences, etc. Many parsers, translation systems, and extraction systems rely on such segments to process the data.
As noted in Chapter 1, text segmentation is a necessary prerequisite for the multi-dimensional classification method proposed by Pan et al. [33, 49]. In order to characterize statements made in the literature along the five dimensions introduced by Wilbur et al. [53], we first have to break the sentences into fragments and then classify these fragments along the five dimensions.

In Section 2.2, we first describe an annotation scheme used to manually categorize sentence fragments along the five dimensions: Focus, Polarity, Certainty, Evidence, and Trend. We then introduce, in Section 2.3, the classifier proposed by Pan et al. [33, 49], used to automatically categorize sentence fragments.

## 2.2 Fragment Annotation

In order to characterize sentence fragments within the biomedical literature, Wilbur et al. [53] proposed five qualitative dimensions defined as follows:

- **Focus**: Indicates the type of the information given by a fragment, which can be Scientific findings, Methodology of a scientific experiment, or Generic information such as a general state of knowledge.

- **Evidence**: Denotes whether the fragment statement is supported by evidence. The types of evidence are defined as follows:
  - $E0$: No indication of evidence or a lack of evidence.
  - $E1$: A claim of evidence but no explicit reference is provided.
  - $E2$: Explicit reference is provided to other publications.
  - $E3$: Evidence is directly given within a fragment.
• Polarity: Indicates whether the statement is stated *Positively* or *Negatively*.

• Trend: Indicates whether an *Increase* or *Decrease* in a specific phenomenon is stated.

• Certainty: Distinguishes the certainty degree regarding the validity of the assertion stated in a fragment, on a scale of 0-3. The lowest degree 0 represents *Complete uncertainty* while the highest degree, 3, represents *Complete certainty*. Additionally, degrees 1 and 2 represent *Low certainly* and *High likelihood* respectively.

Sentence fragment annotation along each of the dimensions, except the *Focus* and *Evidence* dimensions, is independent from other dimensions. The annotation along these two dimensions is correlated [33] and must be performed simultaneously. Thus, in this thesis, we only concentrate on fragment annotation along the *Focus* and *Evidence* dimensions.

To illustrate the annotation scheme, we give an example of fragment annotation along the *Focus* and *Evidence* dimensions:

*Although neuropathologic studies of autism are limited, **1GE1 reports of Purkinje and granule cell loss in Cblm (16) also suggest overlap with this neonatal infection paradigm. **2SE2* 

The sentence is fragmented into two parts to reflect the changes in the *Focus* and the *Evidence*. The first fragment makes a claim only about the number of studies, not their scientific content. Therefore the *Focus* is *Generic* rather than *Scientific*. Additionally, as there is reference to “studies” but no explicit citation, the *Evidence*
level is E1. The second fragment does discuss an explicit phenomenon and provides
a citation, hence the fragment is viewed as *Scientific*, with evidence level E2.

In Section 2.3, we describe the machine learning algorithm proposed by Pan *et
al.* [33, 49] that automatically annotates fragments along the *Focus* and *Evidence*
dimensions.

### 2.3 Fragment Classification

The goal of fragment classification is to automatically categorize each fragment along
all the five dimensions. However, as previously noted, in this thesis, we only concen-
trate on fragment classification along the *Focus* and *Evidence* dimensions. The task
of automatic fragment annotation can be divided into three subtasks. First, training
examples are manually annotated by human experts under the annotation scheme [53]
explained in Section 2.2. Second, the Maximum Entropy classifier proposed by Pan
*et al.* [33, 49] is trained on the manually annotated fragments. Third, fragments are
automatically annotated along the *Focus* and *Evidence* dimensions using the trained
classifier.

Prior to fragment classification along the *Focus* and *Evidence* dimensions, it is
necessary to preprocess the text in order to map it into a format that is interpretable
by our classification method. In Section 2.3.1, we briefly explain the preprocessing
scheme proposed by Pan [33] used to represent raw text as a vector of term weights.
In Section 2.3.2, we then review the Maximum Entropy model proposed by Pan *et
Al.* [33, 49] used to classify fragments along the *Focus* and *Evidence* dimensions.
2.3.1 Text Preprocessing

The preprocessing scheme proposed by Pan [33] maps raw text into a vector of term weights. Each term weight is binary (0 or 1) and indicates the presence or absence of a term that is critical in deciding category boundaries along all the five dimensions. Since, this thesis only concentrates on the Focus and Evidence dimensions, we only explain the preprocessing procedures along these two dimensions. Thus the raw text is represented as two term weight vectors, each of which consists of the weights of the terms defined for the Focus and Evidence dimensions, respectively. In the classification scheme proposed by Pan [33], individual words and statistical phrases (a sequence of $n$ consecutive words occurring with certain frequencies in the text, also known as $n$-grams) are used as terms. Thus, first, individual words and statistical phrases are formed from the text. Second, reduction of the term space is applied to obtain the final terms whose weights are used to form the two feature vectors. In the following, we briefly describe the two steps.

**Term Formation**

In this step, we first break the raw text into a sequence of tokens. We then assign part-of-speech (PoS) tags such as *noun*, *verb* or *preposition*, to each token. The Medpost tool [50] is used to perform tokenization and part-of-speech tagging.

Then the word sequences of length up to 3 (including single words, bigrams and trigrams) are generated. Only the word sequences that occur in the text more than a minimum number of times (2 times) are then selected.
**Term Space Reduction**

This step chooses which terms, among all terms formed in the previous step, should be used to form the two feature vectors for the *Focus* and *Evidence* dimensions. Reducing the dimensions of the term space increases the efficiency of the training and test processes in classification.

Reduction of the term space involves stop word removal, removal of terms based on their part-of-speech tags, removal of specific biomedical terms and term selection based on the relationship between individual terms and specific categories:

- **Stop Word Removal:** High frequency words, which do not carry much semantic value, are removed according to a list of common stop words. Pan [33] defined a different set of stop words for each dimension.

- **Term Removal based on Part-of-Speech (PoS) tags:** Pan [33] studied the effect of words in determination of the category level of a sentence fragment based on their PoS tags. For example, to introduce a methodology or experimental results, the author typically uses the past tense. Thus if the PoS tags associated with a term has tense “past”, then that term may convey information about the *Evidence* in a fragment. Table 2.1 defines the criteria of term selection based on PoS tags for each dimension. The weights of the terms with PoS tags marked as 1 are included in the term weight vector while the weights of those marked with 0 are excluded.

- **Biomedical Term Removal:** From the manually annotated examples, we learned that terms specific to the biomedical domain such as protein or gene
Table 2.1: Selecting terms along the *Focus* and *Evidence* dimensions based on their part-of-speech tags. F denotes *Focus* and E denotes *Evidence*. Terms with PoS tags marked as 1 are selected while those marked with 0 are not selected.

<table>
<thead>
<tr>
<th>POS</th>
<th>F</th>
<th>E</th>
<th>Syntactical Category</th>
<th>POS</th>
<th>F</th>
<th>E</th>
<th>Syntactical Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>1</td>
<td>1</td>
<td>coordinating conjunction</td>
<td>VBN</td>
<td>1</td>
<td>1</td>
<td>participle been</td>
</tr>
<tr>
<td>CS</td>
<td>0</td>
<td>1</td>
<td>subordinating conjunction</td>
<td>VBZ</td>
<td>1</td>
<td>1</td>
<td>3rd person singular <em>is</em></td>
</tr>
<tr>
<td>CSN</td>
<td>0</td>
<td>0</td>
<td>comparative conjunction</td>
<td>VDB</td>
<td>1</td>
<td>1</td>
<td>base <em>do</em></td>
</tr>
<tr>
<td>CST</td>
<td>0</td>
<td>1</td>
<td>complementizer</td>
<td>VDD</td>
<td>1</td>
<td>1</td>
<td>past <em>did</em></td>
</tr>
<tr>
<td>DB</td>
<td>1</td>
<td>1</td>
<td>predeterminer</td>
<td>VDG</td>
<td>1</td>
<td>1</td>
<td>participle <em>doing</em></td>
</tr>
<tr>
<td>DD</td>
<td>0</td>
<td>0</td>
<td>determiner</td>
<td>VDI</td>
<td>1</td>
<td>1</td>
<td>infinite <em>do</em></td>
</tr>
<tr>
<td>EX</td>
<td>1</td>
<td>1</td>
<td>existential</td>
<td>VDN</td>
<td>1</td>
<td>1</td>
<td>participle <em>done</em></td>
</tr>
<tr>
<td>GE</td>
<td>0</td>
<td>1</td>
<td>genitive marker</td>
<td>VDZ</td>
<td>1</td>
<td>1</td>
<td>3rd person singular <em>does</em></td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>1</td>
<td>preposition</td>
<td>VHB</td>
<td>1</td>
<td>1</td>
<td>base <em>have</em></td>
</tr>
<tr>
<td>JJ</td>
<td>1</td>
<td>1</td>
<td>adjective</td>
<td>VHD</td>
<td>1</td>
<td>1</td>
<td>past <em>had</em></td>
</tr>
<tr>
<td>JJR</td>
<td>1</td>
<td>1</td>
<td>comparative adjective</td>
<td>VHG</td>
<td>1</td>
<td>1</td>
<td>participle <em>having</em></td>
</tr>
<tr>
<td>JRT</td>
<td>0</td>
<td>1</td>
<td>superlative adjective</td>
<td>VHI</td>
<td>1</td>
<td>1</td>
<td>infinitive <em>have</em></td>
</tr>
<tr>
<td>MC</td>
<td>1</td>
<td>1</td>
<td>number or numeric</td>
<td>VHN</td>
<td>1</td>
<td>1</td>
<td>participle <em>had</em></td>
</tr>
<tr>
<td>NN</td>
<td>1</td>
<td>1</td>
<td>noun</td>
<td>VHZ</td>
<td>1</td>
<td>1</td>
<td>3rd person singular <em>has</em></td>
</tr>
<tr>
<td>NNP</td>
<td>1</td>
<td>1</td>
<td>proper noun</td>
<td>VVB</td>
<td>1</td>
<td>1</td>
<td>base form lexical verb</td>
</tr>
<tr>
<td>NNS</td>
<td>1</td>
<td>1</td>
<td>plural noun</td>
<td>VVD</td>
<td>1</td>
<td>1</td>
<td>past tense lexical verb</td>
</tr>
<tr>
<td>PN</td>
<td>1</td>
<td>1</td>
<td>pronoun</td>
<td>VVG</td>
<td>1</td>
<td>1</td>
<td>present participle</td>
</tr>
<tr>
<td>PND</td>
<td>0</td>
<td>1</td>
<td>determiner as pronoun</td>
<td>VVI</td>
<td>1</td>
<td>1</td>
<td>infinitive lexical verb</td>
</tr>
<tr>
<td>PNG</td>
<td>0</td>
<td>1</td>
<td>genitive pronoun</td>
<td>VVN</td>
<td>1</td>
<td>1</td>
<td>past participle</td>
</tr>
<tr>
<td>PNR</td>
<td>0</td>
<td>0</td>
<td>relative pronoun</td>
<td>VVZ</td>
<td>1</td>
<td>1</td>
<td>3rd person singular</td>
</tr>
<tr>
<td>RR</td>
<td>1</td>
<td>1</td>
<td>adverb</td>
<td>VVNJ</td>
<td>1</td>
<td>1</td>
<td>prenominal past participle</td>
</tr>
<tr>
<td>RR</td>
<td>1</td>
<td>1</td>
<td>comparative adverb</td>
<td>VVGJ</td>
<td>1</td>
<td>1</td>
<td>prenominal present participle</td>
</tr>
<tr>
<td>RRT</td>
<td>0</td>
<td>1</td>
<td>superlative adverb</td>
<td>VVGN</td>
<td>1</td>
<td>1</td>
<td>nominal gerund</td>
</tr>
<tr>
<td>SYM</td>
<td>1</td>
<td>0</td>
<td>symbol</td>
<td></td>
<td>1</td>
<td>1</td>
<td>left parenthesis</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>0</td>
<td>infinitive marker</td>
<td></td>
<td>0</td>
<td>0</td>
<td>right parenthesis</td>
</tr>
<tr>
<td>VM</td>
<td>1</td>
<td>1</td>
<td>model</td>
<td></td>
<td>0</td>
<td>0</td>
<td>comma</td>
</tr>
<tr>
<td>VBB</td>
<td>1</td>
<td>1</td>
<td>base <em>be, am, are</em></td>
<td></td>
<td>0</td>
<td>0</td>
<td>end-of-sentence period</td>
</tr>
<tr>
<td>VBD</td>
<td>1</td>
<td>1</td>
<td>past <em>was,were</em></td>
<td></td>
<td>0</td>
<td>0</td>
<td>dashes, colons</td>
</tr>
<tr>
<td>VBG</td>
<td>1</td>
<td>1</td>
<td>participle <em>being</em></td>
<td></td>
<td>0</td>
<td>0</td>
<td>left quote</td>
</tr>
<tr>
<td>VBI</td>
<td>1</td>
<td>1</td>
<td>infinitive <em>be</em></td>
<td></td>
<td>0</td>
<td>0</td>
<td>right quote</td>
</tr>
</tbody>
</table>
names are not important in determination of the Evidence in a sentence fragment. Biomedical term removal is applied only for selection of informative terms along the Evidence dimension.

- **Term Selection:** A term selection function, Chi-square, is used to measure how predictive a term is for certain categories along each dimension. Highly scoring terms are selected.

The next task in fragment categorization is to build a classifier that is capable of automatically annotating sentence fragments along the Focus and the Evidence dimensions.

### 2.3.2 Maximum Entropy Model for Fragment Classification

A variation on the Maximum Entropy model was proposed by Pan et al. [33, 49] to classify sentence fragments along the Focus and Evidence dimensions simultaneously. Within the Maximum Entropy framework [30], the goal is to construct the model, the conditional probability $Pr(C|F)$, where $F$ is a text fragment, and $C$ is a random variable denoting the possible category label. Such a model assumes nothing about the probability distribution associated with the training examples other than what is observed.

This statistical modeling has two steps. The first step is to find useful features representing fragments, while the second step is to incorporate these features into the model. Let $f_i$ denotes a feature function, which expresses some useful statistics of the training examples. The conditional distribution $Pr(C|F)$ must satisfy the following
constraint:

\[
\frac{1}{|D|} \sum_{F \in D} f_i(F, C(F)) = \sum_{F \in D} p(F) \sum_{c \in \mathcal{C}L} p(c|F) f_i(c, F),
\]

where \( |D| \) denotes the number of training examples, \( F \) denotes a sentence fragment, \( \mathcal{C}L \) denotes the set of all possible classes, \( c \) denotes one class, and \( C(F) \) is the true class of sentence fragment \( F \). The left part of Equation 2.1 represents the expected value of the feature \( f_i \) derived from training examples, while the right part represents the expected value of the feature \( f_i \) calculated from the model, \( Pr(C|F) \).

Among all the conditional probability distributions, \( P = Pr(C|F) \), that satisfy the constraints in Equation 2.1, the task is to find the optimal distribution, \( Pr^*(C|F) \), that maximizes the conditional entropy defined as:

\[
H(P) = - \sum_{F \in D} \sum_{c \in \mathcal{C}L} p(F) p(c|F) \log p(c|F).
\]

To classify a sentence fragment, we first train a classifier on the training examples to which \textit{Focus-Evidence} tags are assigned. Considering the \textit{Focus} dimension, there are 7 possible values: \textit{Scientific}, denoted as S; \textit{Generic}, denoted as G; and \textit{Methodology}, denoted as M; and their combinations, denoted as MS, MG, SG, and MSG. Moreover, the \textit{Evidence} of a fragment can have 4 possible values: \( E0, E1, E2 \), or \( E3 \). Hence, there are 28 \((4 \times 7)\) possible tag pairs.

For each unseen sentence fragment \( F \), we then define a binary feature value as follows:

\[
f_{ij}(F); \quad 1 \leq i \leq 7; 1 \leq j \leq 4.
\]

If \( F \) is tagged by the \( i \)\textsuperscript{th} \textit{Focus} value and the \( j \)\textsuperscript{th} \textit{Evidence} value, then the feature value, \( f_{ij}(F) \) should satisfy the condition:

\[
f_{ij}(F) = 1;
\]
otherwise:

\[ f_{ij}(F) = 0. \]

We then constrain the expected value of each feature, \( f_{ij} \), calculated from the model, \( Pr(C|F) \), to be the same as that derived from the training examples:

\[ E_{\text{training}}(f_{ij}) = E_p(f_{ij}). \]

Finally, the class label with the largest value under the conditional probability distribution is chosen as a final class label for the fragment. A full description of the Maximum Entropy model can be found in Pan’s thesis \[33\].

In Section 2.4, we discuss Shannon’s entropy \[36\] in detail since it is a central concept to this thesis work. So far, we have stated that for each sentence fragment, there is a probability distribution over all the possible class labels given that fragment. Shannon’s entropy is used as a measure of uncertainty reflected in these probability distributions. In our proposed sentence fragmentation algorithm, we further use this measure to find the optimal sentence fragments.

2.4 Entropy and Certainty

In this section, we describe the concept of entropy and how it can be used to measure the uncertainty reflected in a probability distribution. In Sections 2.4.1 and 2.4.2, we then explain basic properties of the entropy, further used in our proposed algorithm in order to find the optimal sentence fragments.

In the area of Information Theory, entropy is a measure of the uncertainty reflected in a probability distribution associated with a random variable. Let \( X \) be a discrete
random variable with \( q \) outcomes \( \{ x_i : i = 1, \ldots, q \} \), each of which has the probability mass function \( p(x_i) \). Thus given a discrete probability distribution:

\[
P = \{ p(x_1), p(x_2), \ldots, p(x_q) \},
\]

the entropy \( H(P) \) is defined as:

\[
H(P) = -\sum_{i=1}^{q} p(x_i) \log p(x_i). \tag{2.2}
\]

A low entropy value indicates greater certainty inherent in a probability distribution, \( P \), while a large entropy value indicates lower certainty reflected in \( P \).

Consider a fair coin toss. The entropy values correspond to the uncertainty that one has about the outcome (heads or tail) of the flip of a coin. For instance, on any given flip of a fair coin, it is equally likely to have one of the two outcomes (heads or tails). In other words, the outcome could be heads or tails with probability \( \frac{1}{2} \) each; Thus:

\[
X = \begin{cases} 
  x_1 & \text{heads, } p(x_1) = \frac{1}{2}; \\
  x_2 & \text{tail, } p(x_2) = \frac{1}{2},
\end{cases}
\]

and the entropy \( H(P) \) is then calculated as follows:

\[
H(P) = -\frac{1}{2} \log \left(\frac{1}{2}\right) - \frac{1}{2} \log \left(\frac{1}{2}\right) = 1.
\]

The entropy of a fair coin toss is 1 bit, which shows that the result of each toss of the coin delivers 1 bit of information. This is the situation of maximum uncertainty as it is most difficult to predict the outcome of the next toss.

However, considering an unfair coin, the probability of one of the sides is higher than that of the other. Hence, there is less uncertainty compared to a fair coin, and observing the result of tossing a slightly biased coin must produce less than 1 bit of
information. For instance, if the probability of heads is $\frac{3}{4}$, then the entropy is:

$$H(P) = -\frac{3}{4}\log\left(\frac{3}{4}\right) - \frac{1}{4}\log\left(\frac{1}{4}\right) = 0.8.$$ 

We can conclude that a low entropy value indicates greater certainty inherent in a probability distribution, while a large entropy value indicates lower certainty reflected in a probability distribution.

### 2.4.1 The Maximum and Minimum values of the Entropy

As discussed in Section 2.4, considering a random variable $X$ with $q$ outcomes $\{x_i : i = 1, \ldots, q\}$, the probability distribution associated with $X$ forms a vector of probabilities defined as follows:

$$P = \{p(x_1), p(x_2), \ldots, p(x_q)\}.$$ 

The probabilities must be real numbers between zero and one, and must sum to one:

$$0 \leq p(x_i) \leq 1;$$

$$\sum_{i=1}^{q} p(x_i) = 1.$$ 

The entropy of a probability distribution takes on its minimum value, zero, when one of the probabilities is one and the others are zero. The minimum entropy value reflects the most certainty inherent in a probability distribution. On the other hand, it is proven that an entropy takes on its maximum value, $\log(q)$, when all the probabilities are the same [36], that is, when the probability distribution is uniform. The maximum entropy value reflects the most uncertainty inherent in a probability distribution.
2.4.2 Joint Entropy

So far we discussed measuring the uncertainty associated with a probability distribution of a single random variable. We now extend the definition to a pair of random variables. The joint entropy is a measure of the uncertainty embedded in the joint distribution $P = Pr(X, Y)$ of a pair of discrete random variables $(X, Y)$ and is defined as:

$$H(P) = - \sum_{x \in \mathcal{X}D} \sum_{y \in \mathcal{Y}D} p(x, y) \log p(x, y),$$

(2.3)

where $\mathcal{X}D$ and $\mathcal{Y}D$ are the sets of possible values for $X$ and $Y$ respectively. If the two random variables are independent of each other, then for each pair of random variables, $(X, Y)$, the joint probability distribution is given by:

$$P(X, Y) = P(X)P(Y),$$

thus the joint entropy have the following form:

$$H(P) = - \sum_{x \in \mathcal{X}D} \sum_{y \in \mathcal{Y}D} p(x)p(y) \log(p(x)p(y)).$$

Similar to the entropy, the joint entropy is an expression of uncertainty inherent in a joint probability distribution. The minimum joint entropy occurs when one of the joint probability values is equal to one while the others are all equal to zero.

2.5 Related Work

Segmentation has been applied to many tasks, although different from the task addressed in this thesis work. We discuss each task in detail in Sections 2.5.1, 2.5.2, and 2.5.3. We then discuss the applications that are most related to our work in Section 2.5.4.
2.5.1 Sentence Segmentation

Sentence boundary disambiguation (SBD) is the task of identifying sentences within a text. SBD is one of the essential problems for many applications of Natural Language Processing such as machine translation [21], parsing [25], and text summarization [2]. A sentence ends with a terminal punctuation, such as period, question mark, or exclamation mark. These end-of-sentence punctuation marks can be ambiguous. For example, a period can denote a decimal point (like “15.28”), an abbreviation (like “Mr.”), or the end of a sentence.

Traditional SBD systems use regular-expressions augmented by a list of abbreviations to find patterns of characters, such as: period, space, and capital letter, which usually occur at the end of a sentence. For example, the Alembic information extraction system [1] contains a UNIX preprocess module which relies on rule sequences and abbreviations for identifying sentence boundaries.

Current research in SBD attempts to apply machine learning techniques to the sentence boundary detection task. Palmer and Hearst [32] developed a sentence boundary labeling algorithm utilizing part-of-speech (PoS) information associated with punctuations. They feed the PoS probabilities of the tokens surrounding the punctuation mark into a feed-forward neural network. The network’s output value indicates the role of the punctuation mark.

The goal of all the above sentence segmentation applications is to divide the text into sentences. However, as noted in Section 2.2, in the work we present here, the segmentation is performed at the fragment level, since the class labels along each dimension can change mid-sentence. Additionally, in sentence segmentation, the sentences boundaries are detected based on lexical analysis, while, in our fragmentation
method, the fragments boundaries are identified by using both semantic and syntactic analysis. In other words, in our fragmentation method, a break can occur where there is a change in any of the Focus or the Evidence dimensions and this change can be identified based on both semantic and syntactic of a sentence. However, segment boundaries are identified using multiple segmentation cues such as lexical, syntactic, pattern, or their combination.

2.5.2 Intra-Sentence Segmentation

A critical problem in machine translation is syntactically analyzing sentences. The amount of time and space typically required by such analysis is nonlinearly proportional to the length of the sentence [11, 17]. For long sentences, the amount of time and space typically required can become prohibitively large. To address this problem, intra-sentence segmentation is performed, breaking long sentences into a few segments, each of which is then analyzed separately.

Several machine translation systems used intra-sentence segmentation to find the proper segmentation positions inside long sentences. Kim et al. [22] used support vector machines (SVMs) to segment long English sentences in an English-Korean translation system. They first segment sentences used as training examples by finding candidates for segmentation positions in those sentences. A trained SVM classifier is then used to predict the optimal segmentation positions for unseen sentences. In their proposed method, words and punctuations including: preposition, colon, double quotes, relative pronoun, coordinate, relative adverb, conjunction, relative adjective, and a comma are likely to be segmentation positions.

In an English-Chinese machine translation system, Li et al. [25] tried to segment...
long English sentences based on partition pattern information constructed manually. Kim and Ehara [21] used a similar approach to solve the segmentation problems in a Japanese-English translation system. All of these methods were based on segment pattern sets built manually.

Similar to sentence segmentation systems, intra-sentence segmentation systems use lexical analysis in order to break sentences into fragments and do not consider different dimensions and the changes associated with them.

2.5.3 Topic Segmentation

Topic segmentation is the task of determining the positions at which topics change in a stream of text. Hearst’s TextTiling algorithm [15] is a technique for topic segmentation. The algorithm first divides the text into a sequence of relevant tokens (blocks). Adjacent blocks are compared for similarity based on vocabulary overlap, and similarity values are assigned to pairs of adjacent blocks. As a result, a sequence of similarity values is generated. Then each segmentation point between the blocks is given a score computed based on the difference between the similarity value of the left and the right blocks. Finally, the system selects topic segment boundaries as the segmentation points with the largest score.

Salton et al. [42] proposed an algorithm that forms a graph whose nodes are paragraphs in the text, and whose edges are labeled with the similarity score between those paragraphs. The edge is drawn between two paragraph nodes if there is a semantic relation between them. The paragraphs that are connected to many other paragraphs with a similarity score above a particular threshold are then selected as the ones that are likely to contain topics discussed in many other paragraphs. Paragraphs
that were connected by high similarity score to a group of paragraphs, which were also connected to each other by high similarity, but connected by low similarity to the preceding paragraph, were considered the start of a segment.

Barzilay and Elhadad [2] have shown that Lexical chains are very useful to detect the main topics in text. They examine relationships such as synonym, hypernym, and antonym derived from the WordNet thesaurus [13], and also repetitions of words. They then group words together based on the semantic relations between them to build lexical chains.

Similar to our fragmentation method, all the above topic segmentation applications take into account both semantic constraints and syntactic relations in order to divide a text into topical segments. Moreover, in topic segmentation, each topical statement discusses one certain topic. However, in our proposed fragmentation algorithm, a fragment does not necessary discuss a certain topic. Each fragment can be categorized along several dimensions, each of which addresses a certain type of information.

2.5.4 Related Applications

Apart from the multi-dimensional classifier proposed by Pan et al. [33, 49], our proposed fragmentation method can be used as a prerequisite for any categorization system that aims to categorize text along different dimensions, each of which addresses a different type of information.

For instance, our proposed fragmentation algorithm can be used in sentiment analysis [39]. Sentiment analysis is an important research area in text mining. There are opinions and feelings expressed in a text. The sentiment found within these opinions can
be *positive* or *negative*. The goal of sentiment analysis is often to classify the *polarity* (*positive* or *negative*) of a given text. Within sentiment analysis classification, there are two main approaches. Lexical approaches which focus on building dictionaries of pre-tagged words [18], and machine learning approaches [35] which extend text classification models. Previous sentiment analysis systems have tackled the problem at different levels, from the document level [35], sentence level [34] as well as phrase level [6].

Considering the phrase level, the machine learning techniques are first trained with a set of phrases that have been manually labeled as positive or negative. A trained classifiers can then come up with a probability of a new phrase being either positive or negative. Our proposed fragmentation method can be used to automate the process of breaking sentences into phrases. In this case, the categories we have defined in Section 2.2 should be replaced by the sentiment categories.
Chapter 3

Overview of Automated Biomedical Text Fragmentation

As described in Section 2.3, the multi-dimensional fragment classifier proposed by Pan et al. [33, 49] automatically tags sentence fragments along five dimensions: Focus, Evidence, Certainty, Polarity and Trend. In order to categorize sentence fragments in biomedical text, we first need to break sentences into fragments and then use the multi-dimensional fragment classifier to categorize each fragment.

In this chapter, we give an overview of automated biomedical text fragmentation, which can automatically process raw text and break sentences into fragments. In Section 3.1, we provide a short introduction to automated biomedical text fragmentation, which is followed by a more detailed description of the subtasks of automated fragmentation: fragmentation generation and fragmentation scoring.
3.1 Automated Biomedical Text Fragmentation: An Introduction

The text we discuss throughout this thesis consists of documents, where each document is seen as a set of sentences. Each sentence, $S$, consists of a sequence of tokens, $T_1, ..., T_n$, (the most distinguishing terms along either the Focus or the Evidence or both dimensions formed as explained in Section 2.3.1), and is defined as follows:

$$S = < T_1, ..., T_n >,$$

where $n$ is the number of tokens in the sentence, and $T_i$ is the $i$th token in the sentence.

The ultimate goal of automated biomedical text fragmentation is to automatically break all the sentences within the given text into fragments, each of which defined as follows:

$$F_{b_i}^{e_i} = < T_{b_i}, T_{b_i+1}, ..., T_{e_i} >; \quad 1 \leq b_i \leq e_i \leq n \quad \text{and} \quad i \in \{1, 2, ..., m\},$$

where $T_{b_i}$ is the first token in the $i$th fragment and $T_{e_i}$ is the last token in that fragment. A sequence of one or more fragments is called fragmentation and is defined as follows:

$$Frag(S) = < F_{b_1}^{e_1}, F_{b_2}^{e_2}, ..., F_{b_m}^{e_m} >; \quad b_1 \leq e_1 \leq b_2 \leq e_2 \leq ... \leq b_m \leq e_m,$$

where $m$ is the number of fragments within the fragmentation and $F_{b_i}^{e_i}$ is the $i$th fragment within that fragmentation. From the three definitions 3.1, 3.2 and 3.3, we can conclude that $b_1 = 1$ and $e_m = n$, which is the length of a sentence.

For a sentence consisting of $n$ tokens, there exist $n-1$ points at which one can break the sentence, that is, between each two consecutive tokens. All these fragmentation points create a set having $n-1$ elements. Each element in this set is an index.
of a token before which a break can occur. The number of subsets of this set is $2^{n-1}$. The empty subset corresponds to the case when we leave a sentence whole (unfragmented). We can say that, for a sentence consisting of $n$ tokens, there exist $2^{n-1}$ possible fragmentations. For example, consider the following sentence consisting of 4 tokens:

$S = < T_1, T_2, T_3, T_4 >$;

the 8 ($2^4-1$) possible fragmentations for $S$ are as follows:

$Frag_1(S) = < F^4_1 >,$
$Frag_2(S) = < F^1_1, F^4_2 >,$
$Frag_3(S) = < F^2_1, F^4_3 >,$
$Frag_4(S) = < F^3_1, F^4_4 >,$
$Frag_5(S) = < F^1_1, F^2_2, F^4_3 >,$
$Frag_6(S) = < F^1_1, F^3_2, F^4_4 >,$
$Frag_7(S) = < F^2_1, F^3_3, F^4_4 >,$
$Frag_8(S) = < F^1_1, F^2_2, F^3_3, F^4_4 >.$

Each fragment within the generated fragmentations is then labeled with class tags based on probabilistic classification described in Section 2.3.2. For each fragment, the proposed classifier in Section 2.3.2 produces a probability distribution over all the possible class labels given that fragment. In the case that the classifier can reach a clear decision, that is, the probability of assigning a specific class label to a fragment is high, the distribution is skewed toward a certain class. In contrast, when the distribution is uniform, the classifier cannot reach a decision. Thus, there is no clear
unique class tag that the classifier can assign.

Among all the generated fragmentations, our automated fragmentation algorithm aims to select only one fragmentation. The strategy used in our approach is that for each fragment within the selected fragmentation, the probability distribution over all the possible class labels given that fragment must be the most skewed.

The task of automated fragmentation can be divided into two subtasks. First, a sentence is fragmented into all its possible fragmentations. Second, a score is assigned to each generated fragmentation based on the certainty reflected in the probability distributions over all the possible class labels given each fragment within that fragmentation. The fragmentation with the highest score is then selected.

Before applying the proposed algorithm to each sentence, we need to preprocess the text in order to prepare the data for our proposed learning algorithm. In Section 3.1.1, we first describe the preprocessing step. In Section 3.1.2 and 3.1.3, we then study the two subtasks of automated fragmentation in more detail.

### 3.1.1 Text Preprocessing

As explained in Section 2.3, fragments classification is a major step in our proposed algorithm. Additionally, as previously noted, in this work, we only consider classification along the *Focus* and *Evidence* dimensions. Hence, as described in Section 2.3.1, we map the raw text into two term vectors. One term vector represents the text along the *Focus* dimension and contains terms conveying information regarding the *Focus* of the text, while the other represents text along the *Evidence* dimension using terms that are important for the determination of the *Evidence* level of a text. We call the terms in the first vector *informative terms* along the *Focus* dimension and the ones
in the second vector informative terms along the Evidence dimension.

The preprocessing procedures are performed separately for each of the two dimensions. As described in Section 2.3.1, the preprocessing procedure is divided into two steps: term formation and term space reduction. Term formation refers to generating individual words and statistical phrases from the text, while term space reduction refers to removing those terms that do not convey much information regarding the Focus or the Evidence of fragments.

As discussed in Section 2.3.1, for the term formation step, we consider three types of preprocessing: tokenization, part-of-speech tagging and statistical phrase generation. Then reduction of the term space is applied by removing stop words and removing terms based on their part-of-speech tags (Table 2.1). Additionally, for the Evidence dimension, we remove terms specific to the biomedical domain such as protein or gene names which are not important in determining the Evidence of a sentence fragment. Furthermore, a term selection function, Chi-square, is used to select terms that have a critical role in predicting class labels along the Focus and the Evidence dimensions.

The following sections describe the subtasks of automated fragmentation: fragmentation generation and fragmentation scoring.

### 3.1.2 Fragmentation Generation

In automated fragmentation, the preliminary task is to generate all possible fragmentations within a sentence. As previously noted, for a sentence consisting of $n$ tokens, there exist $2^{n-1}$ possible fragmentations. For example, 8 possible fragmentations for a sentence consisting of 4 tokens are shown in Figure 3.1.
However, among all possible fragmentations, only the ones satisfying the three following constraints are of interest:

- All the fragments within these fragmentations must contain at least two words.
- All the fragments within these fragmentations must contain at least one informative term along each of the **Focus** and **Evidence** dimensions.
- Any two consecutive fragments within these fragmentations differ in their class labels, along at least one dimension. The reason is that the annotation guidelines defined in Section 2.2 that we use in order to automate fragmentation specify the same fragmentation condition.

We call these fragmentations, *informative fragmentations*. To generate informative
fragmentations for each sentence, first, all the possible fragmentations satisfying the first two criteria are produced for the sentence, that is, all the fragments within the fragmentations must contain at least two words and at least one informative term along each of the Focus and Evidence dimensions. Second, the maximum entropy model described in Section 2.3.2 is applied to all the fragments within each fragmentation to predict their associated class labels. Last, fragmentations with different class labels along each dimension, for any two consecutive fragments within them are recognized as informative fragmentations.

Suppose, for example, that among the 8 fragmentations shown in Figure 3.1, only fragmentations 2 and 8 satisfy the first criterion, that is, all the fragments within them contain at least two words. Additionally, suppose that three fragments $F_2^2, F_3^4$ and $F_1^4$ within fragmentations 2 and 8 also satisfy the second constraint, that is, they consist of at least one informative term along each of the Focus and Evidence dimensions. Considering the third criterion, we can say that since the class labels assigned to $F_2^2$ and $F_3^4$ are two different labels $C_1$ and $C_2$ respectively, fragmentation 2 is recognized as a possible informative fragmentation. Furthermore, fragmentation 8 only consists of a single fragment; therefore it is another possible informative fragmentation.

In summary, for each sentence, there can be multiple informative fragmentations. In the following section, we introduce a score function that aims to select an optimal fragmentation among all the possible informative fragmentations.

### 3.1.3 Fragmentation Scoring

Section 3.1.2 discussed the generation of all informative fragmentations within a sentence. The next task is to define a score function that can assign a score to each
possible informative fragmentation based on the certainty reflected in the probability distributions over all the possible class labels, given each fragment within the fragmentation. The fragmentation with the highest score is then selected.

In this section, we start by measuring the certainty reflected in the probability distribution over all possible class labels given a single fragment. We then extend the idea to the composition of more than one fragment.

Let $\mathcal{CL} \equiv (c_1, ..., c_l)$ be a set of class labels along the Focus and Evidence dimensions. Each element of $\mathcal{CL}$ is a possible class label that can be assigned to each fragment within a fragmentation. The probability over all the elements of $\mathcal{CL}$ given each fragment $F$ is defined as the following conditional probability distribution:

$$P(C_F) = Pr(C = c_i|F) \ ; \forall c_i \in \mathcal{CL}, i = \{1, 2, ..., l\}$$

where $F$ represents a single fragment, and $C$ denotes a discrete random variable that can take any possible element of $\mathcal{CL}$. As described in Section 2.3.2, each conditional probability term, $p(c_i|F)$, in the above distribution can be calculated using the probabilistic classifier proposed by Pan et al. [33, 49]. The uncertainty reflected in the probability distribution $Pr(C = c_i|F)$, for all class labels $c_i$, is then calculated in terms of the entropy as discussed in Section 2.4:

$$H(P(C_F)) = - \sum_{c_i \in \mathcal{CL}} p(C_F^{c_i}) \log p(C_F^{c_i}).$$

(3.4)

As explained in Section 2.4.1, in Equation 3.4, the entropy function reaches its maximum value when the distribution $P(C_F)$ is uniform. On the other hand, the entropy function reaches its minimum value when one of the probabilities is 1 and the others are zero.
Now, consider a fragmentation consisting of $m$ fragments as follows:

$$Frag = < F_1, F_2, ..., F_m >.$$  

There is a conditional probability distribution $Pr(C = c_i | F = F_j)$, for all $c_i$ associated with each fragment $F_j$. We assume that assigning possible class labels to fragments are independent events; thus for $m$ independent events of assigning possible class labels to $m$ different fragments, the joint probability distribution can be written as follows:

$$P(C_{F_1}, C_{F_2}, ..., C_{F_m}) = P(C_{F_1}) \times P(C_{F_2}) \times ... \times P(C_{F_m}). \quad (3.5)$$

As discussed in Section 2.4.2, to measure the uncertainty inherent in the above joint distribution, the joint entropy is used:

$$H(P(C_{F_1}, C_{F_2}, ..., C_{F_m})) = -\sum_{c_i \in CL} p(C_{F_1}^{c_i}, C_{F_2}^{c_i}, ..., C_{F_m}^{c_i}) \log p(C_{F_1}^{c_i}, C_{F_2}^{c_i}, ..., C_{F_m}^{c_i}). \quad (3.6)$$

Same as the entropy function, the more uniform the joint probability distribution $P(C_{F_1}, C_{F_2}, ..., C_{F_m})$ is, the larger is the value of joint entropy function. On the other hand, the more skewed this joint probability distribution is, the smaller the joint entropy function value. Thus, we define the score function as:

$$Score(Frag) = -H(P(C_{F_1}, C_{F_2}, ..., C_{F_m})). \quad (3.7)$$

The larger the value of the score function, the more biased the classification decisions for all the fragments are toward specific class labels along the Focus and Evidence dimensions.

As explained in Section 2.4.1, the values of the joint entropy function, when calculated over $m$ elements, range between zero and $\log(m)$ [36]; therefore:

$$-\log(m) \leq Score(Frag) \leq 0.$$
The highest value of the score function corresponds to the minimum value of the joint entropy function, which is zero. The zero value of the joint entropy function expresses the certainty reflected in the joint distribution \( P(C_{F_1}, C_{F_2}, ..., C_{F_m}) \). This certainty shows that each probability distribution over all the possible class labels, given each fragment within a fragmentation, tends to be biased toward specific class labels.

To summarize, for each sentence in the biomedical text, all its possible informative fragmentations are generated. Then among all these informative fragmentations, the one with the highest score is selected.
Chapter 4

Evaluation Measure

In order to evaluate the performance of our proposed algorithm, it is necessary to compare fragmentations created by our algorithm with human fragmentations. We refer to human fragmentations as the ground truth. In Section 4.1, we discuss Precision and Recall as performance measures for sentence fragmentation followed by a discussion of their limitations. In section 4.2, we then introduce a new measure that solves the limitations associated with using Precision and Recall.

4.1 Performance Measures

Both our proposed algorithm and the human experts break a sentence into a sequence of fragments. We use separation marks ‘|’ to show the fragment boundaries. We call these separation marks reference separators in the ground truth fragmentations and proposed separators in the fragmentations resulting from applying our proposed algorithm. A proposed separator that is located at the position of a reference separator is called a correctly positioned separator. By considering separation marks, the format
of a fragmentation of a sentence $S$ into $m$ fragments, $\text{Frag}(S)$, is written as:

$$\text{Frag}(S) = < F_{b_1}^{e_1}, |F_{b_2}^{e_2}|, ..., |F_{b_m}^{e_m}>$$

$$= < T_{b_1}, T_{b_1+1}, ..., T_{e_1}, |T_{b_2}, T_{b_2+1}, ..., T_{e_2}|, ..., |T_{b_m}, T_{b_m+1}, ..., T_{e_m}>$$

where $|$ denotes the fragments boundaries and $F_{b_i}^{e_i}$ is any fragment within the fragmentation.

Fragmentations resulting from our algorithm and from the ground truth fragmentations, may have separation marks inserted at different positions in the fragmentation. To evaluate the performance of our proposed algorithm, we use the metrics Precision, Recall and F-measure [44] as follows:

$$\text{Precision} = \frac{\text{number of correctly positioned proposed separators}}{\text{total number of proposed separators}},$$

$$\text{Recall} = \frac{\text{number of correctly positioned proposed separators}}{\text{total number of reference separators}},$$

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$  

For instance, consider the following two fragmentations for a sentence $S$:

$$\text{Frag}(S) = < F_2^1, |F_4^3|, |F_6^5|, |F_7^8> = < T_1, T_2, |T_3, T_4, |T_5, T_6|, |T_7, T_8>,$$

$$\text{Frag1}(S) = < F_2^1, |F_5^3|, |F_6^8> = < T_1, T_2, |T_3, T_4, T_5, |T_6, T_7, T_8>,$$

where $\text{Frag}(S)$ is the ground truth fragmentation and $\text{Frag1}(S)$ is the proposed fragmentation. Thus, there is only one correctly positioned separator (after $F_1^2$) in
Frag1(S). Additionally, the number of proposed separators is two while the number of reference separators is three. Hence:

\[
\text{Precision} = \frac{1}{2},
\]

\[
\text{Recall} = \frac{1}{3},
\]

\[
\text{F-Measure} = \frac{2 \times \frac{1}{2} \times \frac{1}{3}}{\frac{1}{2} + \frac{1}{3}} = \frac{2}{5}.
\]

However, Precision and Recall have their own limitations. Given two fragmentations, one may be very similar to the reference fragmentation while the other is quite different from it; however, they may both have the same Precision and Recall. For example, consider the following three fragmentations:

\[
\text{Frag}(S) = < F_1^2, |F_3^7, |F_8^{10} > =< T_1, T_2, |T_3, T_4, T_5, T_6, T_7, |T_8, T_9, T_{10} >, \\
\text{Frag1}(S) = < F_1^2, |F_3^8, |F_9^{10} > =< T_1, T_2, |T_3, T_4, T_5, T_6, T_7, T_8, |T_9, T_{10} >, \\
\text{Frag2}(S) = < F_1^2, |F_3^4, |F_5^{10} > =< T_1, T_2, |T_3, T_4, |T_5, T_6, T_7, T_8, T_9, T_{10} >,
\]

where \text{Frag}(S) is the reference fragmentation while \text{Frag1}(S) and \text{Frag2}(S) are the two proposed fragmentations. Since the number of proposed separators and correctly positioned separators in both \text{Frag1}(S) and \text{Frag2}(S) are the same, the precision and the Recall for both have the same values calculated as follows:

\[
\text{Precision} = \frac{1}{2},
\]

\[
\text{Recall} = \frac{1}{2}.
\]

However, for the first fragmentation, \text{Frag1}(S), the number of tokens between the incorrectly positioned separator and its corresponding reference separator is 1, while
this number is 3 for the second fragmentation, \textit{Frag}2(S). Hence, \textit{Frag}1(S) is more similar to the reference fragmentation than \textit{Frag}2(S), while they share the same Precision and Recall.

A misplaced fragment boundary is not always wrong. As long as the fragments within a sentence are correctly classified, it is less critical if a sentence is fragmented a few tokens before or after where the human would have fragmented it. In the next section, we introduce an alternative measure for evaluation that takes into account the relative positions of the misplaced separators (fragments boundaries), and indicates how close they are to the correct separators.

\subsection{Edit-Distance with Move}

One of the simplest sequence comparison measures is the edit-distance \cite{28, 47} defined as the minimum number of operations required to transform one string into another. In this thesis, we use edit-distance to measure the similarity between the resultant fragmentations from our algorithm and the ground truth fragmentations. The edit-distance between two fragmentations \textit{Frag}1(S) and \textit{Frag}2(S) denoted as $ed(\textit{Frag}1(S), \textit{Frag}2(S))$ is defined as the minimum number of insertion and deletion operations needed to transform \textit{Frag}1(S) into \textit{Frag}2(S). Separators are the only characters that can be involved in any operation, since all the other characters and their locations in the fragmentations resulting from applying our algorithm and the ground truth fragmentations are the same. The operations considered are: separator-insertion and separator-deletion. Considering a fragmentation \textit{Frag}(S) = <T_1, T_2, ..., T_i, ..., T_n> consisting of $n$ tokens and 2 separation marks, the operations affecting \textit{Frag}(S) are defined as follows:
A separator-insertion$_{\text{Frag}(S)}(i)$ inserts a separator at index $i$ in $\text{Frag}(S)$ and transforms $\text{Frag}(S)$ into $< T_1, T_2, |...|T_{i-1}, |T_i, ..., |T_n >$.

A separator-deletion$_{\text{Frag}(S)}(n)$ deletes a separator from index $n$ in $\text{Frag}(S)$ and leads to $< T_1, T_2, |...|T_i, ..., T_{n-1}, T_n >$.

There is a unit cost associated with each insertion and deletion. The sum of these costs shows how expensive it is to transform one fragmentation into the other.

By only considering two operations, separator-insertion and separator-deletion, we are not able to discriminate between an almost correct fragmentation and a fragmentation for which the number of its separation marks and their positions are very dissimilar to those in the correct fragmentation. For example, consider the following three fragmentations:

$\text{Frag}(S) = < T_1, T_2, |T_3, |T_4, T_5, T_6, T_7, T_8 >$,
$\text{Frag}1(S) = < T_1, T_2, |T_3, T_4, |T_5, T_6, T_7, T_8 >$,
$\text{Frag}2(S) = < T_1, T_2, |T_3, T_4, T_5, T_6, |T_7, T_8 >$,

where $\text{Frag}(S)$ is the reference fragmentation while $\text{Frag}1(S)$ and $\text{Frag}2(S)$ are two proposed fragmentations. The edit-distance between $\text{Frag}(S)$ and each of $\text{Frag}1(S)$ and $\text{Frag}2(S)$ is calculated as:

$$ed(\text{Frag}1(S), \text{Frag}(S)) = \text{cost}(\text{separator-deletion}_{\text{Frag}1(S)}(5))$$
$$+ \text{cost}(\text{separator-insertion}_{\text{Frag}1(S)}(4)),$$
and:

\[ ed(Frag_2(S), Frag(S)) = cost(separator-deletion_{Frag_2(S)}(7)) \]

\[ + cost(separator-insertion_{Frag_2(S)}(4)). \]

By comparing the number of tokens between the incorrectly positioned separator in each of \(Frag_1(S)\) and \(Frag_2(S)\) and the reference fragmentation, we can say that \(Frag_1(S)\) is more similar to \(Frag(S)\) than \(Frag_2(S)\). However, by using the edit-distance with only two operations, \(separator-insertion\) and \(separator-deletion\), the distances between the two proposed fragmentations and the reference fragmentation are the same.

To solve this problem, we extend the edit-distance by introducing an additional operation, \(separator-move\). The operator \(separator-move_{Frag(S)}(i, j)\) moves the separator from position \(i\) to position \(j\) in \(Frag(S)\). Hence it transforms:

\[ Frag(S) = < T_1, ..., T_{i-1}, |T_i, ..., T_{j-1}, T_j, ..., T_n > \]

into

\[ Frag'(S) = < T_1, ..., T_{i-1}, T_i, ..., T_{j-1}, |T_j, ..., T_n >. \]

In our proposed measure, namely, the edit distance with move, a \(separator-move\) occurs when one separator is deleted in one position and another is inserted in another position.

Shapira and Store [47] considered an edit distance augmented with the substring move operation. Cormode and Muthukrishnan [9] studied the same distance with moves. In all previous work that introduced an additional \(move\) operation into the
edit-distance formulation, every operation had an equal cost. In contrast, in our proposed measure, while separator-insertion and separator-deletion operations have an equal unit cost, we define the cost of the separator-move operation in a way that can discriminate between a fragmentation that is close to the ground truth fragmentation and a fragmentation that is remote from it. For example consider the following reference fragmentation \(Frag(S)\), and the two proposed fragmentations, \(Frag1(S)\) and \(Frag2(S)\), discussed before:

\[
\begin{align*}
Frag(S) &=< T_1, T_2, |T_3, |T_4, T_5, T_6, T_7, T_8 >, \\
Frag1(S) &=< T_1, T_2, |T_3, T_4, |T_5, T_6, T_7, T_8 >, \\
Frag2(S) &=< T_1, T_2, |T_3, T_4, T_5, T_6, |T_7, T_8 >.
\end{align*}
\]

As previously mentioned, \(Frag1(S)\) is more similar to \(Frag(S)\) than \(Frag2(S)\). A single separator-move operation can be used to transform both \(Frag1(S)\) and \(Frag2(S)\) into \(Frag(S)\). The parameters of the separator-move operation in the first transformation are \((5, 4)\), i.e. moving the separator from position 5 to position 4, while these parameters in the second transformation are \((7, 4)\). The difference between the parameters of the separator-move operation \((5 - 4\) and \(7 - 4\), respectively), shows the number of tokens between each of the incorrectly positioned separators in \(Frag1(S)\) and \(Frag2(S)\) and the position of the reference separator. Hence, by defining the cost for operation separator-move as follows, our proposed measure is able to assess how close each fragmentation is to the correct fragmentation:

\[
Cost(\text{separator-move}_{Frag(S)}(p_1, p_2)) = |p_1 - p_2|.
\]  

(4.1)
To calculate the total cost of transforming one fragmentation into another, we have to sum the costs over all the operations.

In the next section, we present an algorithm used to compute the edit-distance with separator-move between two fragmentation.

4.2.1 Algorithm for Calculating Edit-Distance with Move

This section aims at presenting the calculation of our proposed edit-distance measure between two given fragmentations, which includes a cost factor for the different edit operations like separator-insertion, separator-deletion, and separator-move.

Suppose that $Frag(S)$ is a reference fragmentation and $Frag'(S)$ is a proposed fragmentation. As previously noted, except for separators, all the other characters and their locations in the two fragmentations are the same. Hence, the goal is to transform $Frag'(S)$ into $Frag(S)$ by a series of edit operations on separators only.

We first define a single function, $Index$, that takes a fragmentation as an input and returns a list of the indexes following each separator. For example, applying the function $Index$ to the two fragmentations shown in Figure 4.1, forms the following two lists:

\[
Index(Frag(S)) = Ind_{Ref} = [3, 9],
\]
\[
Index(Frag'(S)) = Ind_{Prop} = [2, 5, 10].
\]

For simplicity, we denote the first list, $Index(Frag(S))$ as $Ind_{Ref}$, and the second list, $Index(Frag'(S))$ as $Ind_{Prop}$.

In order to calculate the edit-distance between the proposed and the reference
Ref \quad T_1 | T_2 | T_3 | T_4 | T_5 | T_6 | T_7 | T_8 | T_9 | T_{10}

Prop \quad T_1 | T_2 | T_3 | T_4 | T_5 | T_6 | T_7 | T_8 | T_9 | T_{10}

Figure 4.1: Ref is the reference fragmentation, Frag(S), while Prop is the proposed fragmentation, Frag’(S). Tokens and separators are shown in the two fragmentations.

fragments, we have to compute the cost of transforming Ind_{Prop} into Ind_{Ref}.

Any difference between the two lists Ind_{Prop} and Ind_{Ref} is a result of one of the following mismatches:

- If there is a separator in the reference fragmentation that does not have a corresponding separator at the same position in the proposed fragmentation. Pevzner and Hearst [38] refer to this case as a false negative.

- If there is a separator in the proposed fragmentation that does not have a corresponding separator at the same position in the reference fragmentation. Pevzner and Hearst [38] refer to this case as a false positive.

For instance, consider the two fragmentations shown in Figure 4.1. There are two false negatives occurring at positions 3 and 9 in the reference fragmentation. In addition, the proposed fragmentation introduces three false positives at positions 2, 5, and 10.

We handle false positives and false negatives by the two procedures introduced below:

- The match-move procedure

- The match-insert-delete procedure.
The *match-move* procedure seeks to possibly move each false positive separator to the same position as the reference separator on its right or its left. The only operation permitted in this procedure is the *separator-move* operation. The cost for the *match-move* procedure is the sum of the costs for all its *separator-move* operations, and is computed recursively by breaking the matching problem into a set of smaller matching subproblems. The following three cases are used to generate the possible matching subproblems:

**A)** We match the proposed separator at position $\text{Ind}_{\text{Prop}}[i]$ in $\text{Frag}'(S)$ to the reference separator at position $\text{Ind}_{\text{Ref}}[j]$ in $\text{Frag}(S)$, using a *separator-move* operation. The cost for this match is computed using Equation 4.1 as follows:

$$\text{Cost}(\text{separator-move}_{\text{Frag}'(S)}(\text{Ind}_{\text{Prop}}[i], \text{Ind}_{\text{Ref}}[j])) = |\text{Ind}_{\text{Prop}}[i] - \text{Ind}_{\text{Ref}}[j]|,$$

where $i$ and $j$ are the indexes in $\text{Ind}_{\text{Prop}}$ and $\text{Ind}_{\text{Ref}}$ from which we start matching separators to the end of both lists. We start from the first element in both lists $\text{Ind}_{\text{Prop}}$ and $\text{Ind}_{\text{Ref}}$, that is, when $i$ and $j$ are set to zero. We then move forward one element in both lists $\text{Ind}_{\text{Prop}}$ and $\text{Ind}_{\text{Ref}}$. We use $\text{Ind}_{\text{Prop}}[i+1,..,M]$ and $\text{Ind}_{\text{Ref}}[j+1,..,N]$ to refer to the suffixes of both lists, where $M$ is the number of elements in $\text{Ind}_{\text{Prop}}$ and $N$ is the number of elements in $\text{Ind}_{\text{Ref}}$. The cost of all *separator-move* operations to transform the suffix of $\text{Ind}_{\text{Prop}}$, $\text{Ind}_{\text{Prop}}[i+1,..,M]$, into the suffix of $\text{Ind}_{\text{Ref}}$, $\text{Ind}_{\text{Ref}}[j+1,..,N]$, is then computed. This subproblem is exactly the same as the original matching problem, except that we aim to match the suffixes of the two lists. We can solve this subproblem recursively.

**B)** We move forward one element in the list $\text{Ind}_{\text{Ref}}$, that is, we avoid matching
a reference separator at position \( Ind_{Ref}[j] \) in \( Frag(S) \) to any proposed separators in \( Frag'(S) \). We then need to match the proposed separators from position \( Ind_{Prop}[i] \) to the end of the proposed fragmentation with the reference separators from position \( Ind_{Ref}[j+1] \) to the end of the reference fragmentation. This subproblem is identical to the original matching problem, except that we aim to match the list \( Ind_{Prop}[i, .., M] \), with the list \( Ind_{Ref}[j+1, .., N] \).

\( C' \) This case is similar to Case B except that we move forward one element in the list \( Ind_{Prop} \), that is, we avoid matching a proposed separator at position \( Ind_{Ref}[i] \) in \( Frag'(S) \) to any reference separators in \( Frag(S) \). We then need to match the reference separators from position \( Ind_{Ref}[j] \) to the end of the reference fragmentation with the proposed separators from position \( Ind_{Prop}[i+1] \) to the end of the proposed fragmentation. This subproblem is identical to the original matching problem, except that we aim to match the list \( Ind_{Ref}[j, .., N] \), with the list \( Ind_{Prop}[i+1, .., M] \).

The function \( \text{match-move}(Ind_{Prop}, Ind_{Ref}, i, j) \) returns the minimum cost for matching the list \( Ind_{Prop}[i, .., M] \) with the list \( Ind_{Ref}[j, .., N] \). In other words, the minimum of the following three costs is computed:

- The cost resulting from \( \text{match-move}(Ind_{Prop}, Ind_{Ref}, i+1, j+1) \) + the cost of matching \( Ind_{Prop}[i] \) with \( Ind_{Ref}[j] \) (denoted cost A).

- The cost resulting from \( \text{match-move}(Ind_{Prop}, Ind_{Ref}, i+1, j) \) (denoted cost B).

- The cost resulting from \( \text{match-move}(Ind_{Prop}, Ind_{Ref}, i, j+1) \) (denoted cost C).

The method is bottom-up, that is, starts from calculating the cost for small subproblems and uses those to calculate the cost for larger problems. The pseudo-code for
the match-move procedure is shown in Table 4.1. Lines 6 to 9 in the pseudo-code are related to Case A, while line 11 is related to Case B. Additionally, Case C is presented in line 12.

In the first procedure, the same subproblems might appear in different subproblems. To avoid recalculation of overlapping subproblem results, dynamic programming is used. The cost for unique subproblems are stored in a table. Then, to avoid recomputing, whenever an overlapping subproblem is encountered, one checks whether it has already been solved. If so, the already computed cost that is stored in the table is used.

As can be seen from Table 4.1, each case has two outputs. One output represents the cost of a sequence of separator-move operations. The other output is the new list of the indexes following each proposed separator, called Ind\(_{\text{Prop}}\).

There are two functions, isValidMove and isValidMatching, that we use in the pseudo-code in lines 5, 13, and 16. In the following, we discuss each of them in detail.

**Function isValidMove:**
In Case A, we have to check the validity of any separator-move operation. The function isValidMove is used to determine the validity of moving a proposed separator at position Ind\(_{\text{Prop}}[i]\) in Frag\(_{\text{Prop}}(S)\) to the same position as the reference separator (Ind\(_{\text{Ref}}[j]\)). In order for a move to be valid, there must not exist any reference separator between the reference separator at position Ind\(_{\text{Ref}}[j]\) in Frag\(_{\text{Ref}}(S)\) and the proposed separator at position Ind\(_{\text{Prop}}[i]\) in Frag\(_{\text{Prop}}'(S)\). If the function isValidMove returns false, it means that there is another reference separator closer (the number of tokens between that reference separator and the proposed separator is smaller) to
Table 4.1: The match-move procedure

**Input:**
Ind\_Prop: A list of the break points indexes in the proposed fragmentation.
Ind\_Ref: A list of the break points indexes in the reference fragmentation.
i, j: The indexes in Ind\_Prop and Ind\_Ref respectively, from which we start matching the separators to the end of both lists.

**Output:**
cost: The cost for all separator-move operations.
Ind\_Prop\': The list indicating which separators should be moved in the proposed fragmentation.

Procedure **match-move**(Ind\_Prop, Ind\_Ref, i, j){
1: if i > size-of (Ind\_Prop) or j > size-of (Ind\_Ref) then;
2: return (0, Ind\_Prop);
3: end if
4: costA, costB, costC ← MAXINT;
5: if isValidMove(Ind\_Prop, Ind\_Ref, i, j) then;
6: (costA, Ind\_Prop\' \_i) ← match-move(Ind\_Prop, Ind\_Ref, i + 1, j + 1);
7: costMove ← Cost(separator-move(Ind\_Prop[i], Ind\_Ref[j]));
8: Ind\_Prop\'[i] ← Ind\_Ref[j];
9: costA ← costA + costMove;
10: end if
11: (costB, Ind\_Prop\' \_i) ← match-move(Ind\_Prop, Ind\_Ref, i, j + 1);
12: (costC, Ind\_Prop\' \_i) ← match-move(Ind\_Prop, Ind\_Ref, i + 1, j);
13: if !isValidMatching(Ind\_Prop\' \_i, Ind\_Ref, i, j) then
14: costB ← MAXINT;
15: endif
16: if !isValidMatching(Ind\_Prop\' \_i, Ind\_Ref, i, j) then
17: costC ← MAXINT;
18: endif
19: cost ← min(costA, costB, costC);
20: if cost == costA then
21: Ind\_Prop\' ← Ind\_Prop\' \_i;
22: else if cost == costB then
23: Ind\_Prop\' ← Ind\_Prop\' \_i;
24: else if cost == costC then
25: Ind\_Prop\' ← Ind\_Prop\' \_i;
26: end if
27: return (cost, Ind\_Prop\');
}

46
the proposed separator at position \( \text{Ind}_{\text{Prop}}[i] \) in \( \text{Frag}'(S) \), to which we can match the proposed separator. For example, for the two fragmentations shown in Figure 4.2, with the following two lists of break points indexes in the reference and the proposed fragmentations:

\[
\text{Ind}_{\text{Ref}} = [2, 3, 7, 8]; \quad j \in \{0, 1, 2, 3\},
\]
\[
\text{Ind}_{\text{Prop}} = [4, 6]; \quad i \in \{0, 1\}.
\]

The two following cases are not valid:

- \( \text{match-move}(\text{Ind}_{\text{Prop}}, \text{Ind}_{\text{Ref}}, 0, 0) \).
- \( \text{match-move}(\text{Ind}_{\text{Prop}}, \text{Ind}_{\text{Ref}}, 1, 3) \).

Considering the first case, matching a proposed separator at position 4, \( \text{Ind}_{\text{Prop}}[0] \), in \( \text{Frag}'(S) \) to a reference separator at position 2, \( \text{Ind}_{\text{Ref}}[0] \), in \( \text{Frag}(S) \) is invalid. Similarly, for the second matching case, matching a proposed separator at position 6, \( \text{Ind}_{\text{Prop}}[1] \), in \( \text{Frag}'(S) \) to a reference separator at position 8, \( \text{Ind}_{\text{Ref}}[3] \), in \( \text{Frag}(S) \) is invalid. In both cases, there is another reference separator which is closer to the

\[
\begin{array}{ccccccc}
\text{Ref} & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 \\
\text{Prop} & T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8
\end{array}
\]
proposed separator to which we can match the proposed separator. In our example, the proposed separator at position 4 must be matched to the reference separator at position 3. Moreover, the proposed separator at position 6 must be matched to the reference separator at position 7.

**Function is ValidMatching:**

As previously discussed, in Case B, we avoid matching a reference separator at position $Ind_{Ref}[j]$ in $Frag(S)$ to any proposed separators in $Frag'(S)$. Further matches can then generate the new list of the break point indexes in the proposed fragmentation, called $IndB'_{Prop}$, in which any possible number of break point indexes are replaced by the indexes of the reference separators.

If we compare the suffix of the list $IndB'_{Prop}, IndB'_{Prop}[i, .., M]$, with the suffix of the list $Ind_{Ref}, Ind_{Ref}[j, .., N]$, we see that there might be some false positive separators that were not matched to any reference separator. We refer here to these false positive separators, free proposed separators.

As previously noted, the match-move procedure seeks to possibly move each false positive separator to the same position as the reference separator on its right or its left. If in Case B, our algorithm did not avoid matching the reference separator at position $Ind_{Ref}[j]$ in $Frag(S)$ with any proposed separator in $Frag'(S)$, then one of the free proposed separators might be matched to this reference separator at position $Ind_{Ref}[j]$ in $Frag(S)$. By using the function is ValidMatching, we ensure that there is no free proposed separator from position $IndB'_{Prop}[i]$ to the end of the proposed fragmentation that could be matched to the reference separator at position $Ind_{Ref}[j]$ in $Frag(S)$. 

48
We assign $\text{MAX-INT}$ to the cost of any of the cases in which the function $\text{isValidMatching}$ returns $\text{false}$. Similarly, the $\text{isValidMatching}$ function is used for Case $C$, but with a different set of parameters $(\text{IndC}_\text{Prop}, \text{Ind}_\text{Ref}, i, j)$.

Finally, as previously noted, among Case $A$, Case $B$, and Case $C$, the one that has the minimum cost is of interest. After applying the first procedure on the two fragmentations shown in Figure 4.1, the new list of the indexes following each proposed separator is as follows:

$$\text{Ind'}_{\text{Prop}} = [3, 5, 9],$$

which corresponds to moving a proposed separator at position 2 to position 3 in $\text{Frag}'(S)$ and also moving a proposed separator at position 10 to position 9 in $\text{Frag}'(S)$ to possibly transform the proposed fragmentation into the reference fragmentation. Additionally, the corresponding cost resulting from the first procedure is as follows:

$$\text{Cost}(\text{separator-move}_{\text{Frag}'(S)}(2, 3)) + \text{Cost}(\text{separator-move}_{\text{Frag}'(S)}(10, 9)) = 2$$

So far, the $\text{match-move}$ uses all the possible $\text{separator-move}$ operations to possibly move false positive separators in $\text{Frag}'(S)$ to the position of reference separators. The second procedure, $\text{match-insert-delete}$, completes transforming $\text{Frag}'(S)$ into $\text{Frag}(S)$ by using only two operations $\text{separator-insertion}$ and $\text{separator-deletion}$. The inputs to the $\text{match-insert-delete}$ procedure are the list $\text{Ind'}_{\text{Prop}}$, which is the output of the $\text{match-move}$ procedure, and the list $\text{Ind}_\text{Ref}$. In our example, with the
following $\text{Ind}_{\text{Ref}}$ and $\text{Ind}^\prime_{\text{Prop}}$:

$$\text{Ind}_{\text{Ref}} = [3, 9],$$

$$\text{Ind}^\prime_{\text{Prop}} = [3, 5, 9],$$

in order to complete the transformation, we have to delete 5 from the list $\text{Ind}^\prime_{\text{Prop}}$ which corresponds to deleting the proposed separator at position 5 in the proposed fragmentation.

There is a unit cost associated with each of the separator-deletion and separator-insertion operation. The total cost is the sum of the computed costs for the first ($\text{Cost}(\text{match-move})=2$) and the second procedure ($\text{Cost}(\text{match-insert-delete})=1$), which is 3.

In summary, our proposed algorithm for the edit-distance with move is used to calculate the cost of transforming the proposed fragmentation into the reference fragmentation. In the next chapter, we present the fragmentation results along the Focus and Evidence dimensions, as well as the fragmentation performance.
Chapter 5

Experiments and Results

In this chapter, we first introduce in Section 5.1 the dataset used in our experiments. We then present in Section 5.2 a set of experiments to automatically perform sentence fragmentation followed by a discussion of evaluation measures in Section 5.3. We then present in section 5.4 the results obtained by applying our automated fragmentation algorithm, and evaluate the performance of the proposed algorithm.

5.1 Dataset

This section briefly presents the set of sentences that were annotated by human experts and also the subset to which we apply the automated fragmentation algorithm. We use a subset of a corpus consisting of 10,000 sample sentences collected from Medline abstracts and journal articles. The collected sentences were each previously annotated by three independent biomedical scientists trained using previously defined annotation guidelines [3].

To be included in the subset to which we apply our algorithm, a sentence must
satisfy two main requirements. First, since our automated fragmentation algorithm concentrates only on fragmenting sentences along the Focus and Evidence dimensions, only sentences for which all three annotators assigned the same annotation tags along these two dimensions are used in our experiments. Second, selected sentences must be informative along both the Focus and Evidence dimensions, that is, they must contain informative terms along these two dimensions. Table 5.1, column Sent-FE shows the number of sentences and respective human-annotated fragments for which all three annotators agreed with respect to the Focus and Evidence dimensions. By removing non-informative sentences from the basic dataset, we obtain the dataset Sent-FE-INFO whose statistics are shown in the right column of Table 5.1. This subset, consisting of 1804 sentences, is used in our experiments.

Table 5.1: Dataset information: Sent-FE generated based on the annotation agreement of all three annotators. Sent-FE-INFO contains only informative sentences in Sent-FE.

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>Sent-FE</th>
<th>Sent-FE-INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of sentences</td>
<td>1968</td>
<td>1804</td>
</tr>
<tr>
<td>Number of human-annotated fragments</td>
<td>2010</td>
<td>1844</td>
</tr>
</tbody>
</table>

We refer to the human fragmentations for these 1804 sentences as the ground truth. The following sections describe the experiments and results.

5.2 Automated Fragmentation

For the automated fragmentation experiments, the text consisting of the sentences in the set Sent-FE-INFO must be represented as a vector of informative terms.
We follow the preprocessing procedures discussed in Section 3.1.1. First, from the raw text, discriminating terms along the Focus and Evidence dimensions are formed by individual words and statistical phrases. Second, the term space is reduced by removing stop words, removing terms occurring less than two times, and removing words which do not convey information about the Focus and Evidence of a fragment (Table 2.1). Then the term selection function Chi-square is used to select terms which are predictive for the Focus and Evidence dimensions. In Pan’s work on fragment classification [33], the highest prediction accuracy is obtained when the number of selected terms is between 500 and 700 for the Focus dimension, and between 100 and 200 for the Evidence dimension. Therefore, the top 600 informative terms along the Focus dimension and the top 200 informative terms along the Evidence dimension are selected and used to generate the term vector representing the text.

After the preprocessing step, the preliminary task in our proposed automated fragmentation algorithm is to generate the possible informative fragmentations that satisfy the three constraints discussed in Section 3.1.2, within each sentence. As defined in Section 3.1.2, all the fragments within an informative fragmentation should contain at least two words and at least one informative term along each of the Focus and Evidence dimensions. Additionally, any two consecutive fragments within an informative fragmentation must differ in their class labels.

After generating all the possible informative fragmentations within each sentence, the next task is to assign a score to each of the generated informative fragmentations. As discussed in Section 3.1.3, for a fragmentation, $Frag(S)$, consisting of $m$ fragments, $F_1, F_2, ..., F_m$, the score function is given by:

$$\text{Score}(Frag(S)) = -H(Pr(C_{F_1}, C_{F_2}, ..., C_{F_m})),$$
where \( H \) is the joint entropy function calculated with respect to the joint probability distribution associated with a fragmentation. This joint probability distribution is calculated using Equation 3.5. The value of the above score function, which is based on the joint entropy, indicates how biased the fragments classifications are toward specific class labels along the *Focus* and *Evidence* dimensions.

Finally for each sentence, the fragmentation with the highest score, that is, the one with the lowest joint entropy value, is selected. The classification decisions for the fragments within the selected fragmentation have the highest certainty toward specific class labels along the *Focus* and *Evidence* dimensions.

### 5.3 Performance Evaluation

The performance of the proposed automated fragmentation algorithm is evaluated by comparing the fragmentations generated by our proposed algorithm with fragmentations produced by human annotators. Another way to evaluate the performance of our proposed algorithm is by comparing it against another, baseline, fragmentation algorithm. In the following section, we first describe the specifications of the human fragmentations, to which we refer as the ground truth. We then describe the measures used to evaluate the performance of our algorithm.

#### 5.3.1 Fragmentation by Human Expert

As discussed in Section 2.2, human experts were directed to fragment a sentence at the point where there is a change along any of the five dimensions. However, as noted in Section 3.1, we only consider sentences in which the change occurs along either the
Focus or Evidence dimensions. The distribution of the number of fragments in the resulting fragmentations is shown in Figure 5.1. As can be seen from the figure, the distribution is highly skewed. Most of the sentences have only one fragment which is the sentence itself, while relatively few have two fragments. In other words, among the 1804 sentences, human experts only fragmented 40. Thus, according to the human annotators, for the remaining 1764 sentences, there is no change in the class label along either the Focus or Evidence dimensions throughout the whole sentence.

We first focus our attention on the 40 sentences that were fragmented into two fragments by human experts. We investigated the positive correlation between the length of sentences and the length of each of its individual fragments within these 40 sentences.

The following formula is used to obtain the desired correlation coefficient matrix
between the length of the $k^{th}$ fragments and the length of sentences:

$$R_{L_{F_k}, L_{S}} = \frac{40 \sum_{i=1}^{40} L_{F_{k,i}} L_{S_i} - \sum_{i=1}^{40} L_{F_{k,i}} \sum_{i=1}^{40} L_{S_i}}{\sqrt{40 \sum_{i=1}^{40} L_{F_{k,i}}^2 - (\sum_{i=1}^{40} L_{F_{k,i}})^2} \sqrt{40 \sum_{i=1}^{40} L_{S_i}^2 - (\sum_{i=1}^{40} L_{S_i})^2}},$$

where 40 shows the number of sentences and $L_{F_{k,i}}$ is the length of the $k^{th}$ fragment within the $i^{th}$ sentence; moreover, $L_{S_i}$ is the length of the $i^{th}$ sentence. Both a sentence length and a fragment length are defined as the number of tokens within a sentence or a fragment respectively. For example, for a sentence consisting of two fragments with lengths $L_1$ and $L_2$ respectively, the parameters $\{L_{F_k} : k = 1, 2\}$ and $L_S$ are defined as follows:

$$L_{F_1} = L_1,$$

$$L_{F_2} = L_2,$$

$$L_S = L_1 + L_2.$$

The correlation coefficient in Equation 5.3.1 is used to measure the degree of linear dependence between each of $L_{F_1}$ and $L_{F_2}$, and $L_A$. The resulting correlation matrices after applying Equation 5.3.1 to the 40 sentences are as follows:

$$R(L_{F_1}, L_A) = \begin{bmatrix} 1 & 0.80 \\ 0.80 & 1 \end{bmatrix},$$

$$R(L_{F_2}, L_A) = \begin{bmatrix} 1 & 0.72 \\ 0.72 & 1 \end{bmatrix}.$$
The above correlation matrices show strong linear correlation between the length of sentences and the length of fragments within these sentences. In other words, an increase in the length of a sentence is associated with a proportional increase in the length of the individual fragments within that sentence. We further use this information in the evaluation process to assess the performance of our proposed fragmentation algorithm.

5.3.2 Evaluation Measures

The proposed edit-distance approach from Section 4.1 is used to measure the similarity between resultant fragmentations from our proposed algorithm and those generated by human experts. Each edit-distance value represents the total cost of separator-move, separator-insertion, and separator-deletion that can be used to transform one fragmentation into the other. However, these values do not convey any information about how well our algorithm works compared to other existing fragmentation algorithms.

Another way to evaluate the performance of our algorithm is to compare it with another fragmentation algorithm. Since there is no other algorithms that fragments sentences along the Focus and Evidence dimensions, we use as a baseline for comparison randomly generated fragmentations, which are produced using some properties of the human-annotated fragmentations explained in the previous section.

As shown in Figure 5.1, out of the entire dataset, human annotators only fragmented 40 sentences into two fragments, that is, 3% of the sentences and they leave the rest 97%, that is, 1764 sentences, unfragmented. Thus, for any given sentence, we define the randomization process consisting of two steps as follows:
• **Step1**: Toss a coin with:

\[
Pr(H) = \frac{40}{1804} = 0.03,
\]

\[
Pr(T) = \frac{1764}{1804} = 0.97.
\]

• **Step2**: If the coin comes up tails keep the sentence whole (unfragmented); otherwise fragment the sentence as explained below.

When a sentence is fragmented into two fragments, we have to generate a random number indicating the index of the token at which the fragmentation between the two fragments occurs. This random number is produced using some properties of the human-annotated fragmentations explained in the previous section.

As discussed in the previous section, for those 40 sentences that were also fragmented by human experts, the indexes of the fragmentation points are proportional to the length of the sentences. Figure 5.2 shows the distribution of fragmentation positions normalized by the length of sentences in human-annotated fragmentations. For example, for a sentence consisting of 10 tokens and a fragmentation point at index 6 (that is before the 6th token), the normalized fragmentation position value is \(\frac{6}{10}\). In other words, because all 40 sentences have only 2 fragments, the normalized fragmentation position is the ratio between the length of the first fragment in each sentence and the length of the sentence. Hence, to fragment a sentence, we randomly select a normalized index value from the distribution shown in Figure 5.2. Then by multiplying this number by the length of the sentence, we obtain the index of the token at which we break the sentence.
Figure 5.2: The distribution of the index of the fragmentation point normalized by the length of a sentence, for 40 sentences that were also fragmented by the human annotators.

Finally, the edit-distance values between the randomly generated fragmentations and the human fragmentations are calculated and compared to the edit-distance values measured between fragmentations resulting from our algorithm and the ground truth.

### 5.4 Experimental Results

In this section, we present the results of the experiments described in Section 5.2. The results of selecting the optimal fragmentations within sentences, after generating all their possible informative fragmentations, are examined in Section 5.4.1. We then evaluate the performance of our proposed algorithm in Section 5.4.2. The evaluation is performed based on the approach discussed in Section 5.3.2.
5.4.1 Automated Fragmentation Results

For a sentence consisting of \( n \) tokens, applying the constraints discussed in Section 3.1.2 leads to the generation of a very large number of fragmentations, that is close to \( 2^{n-1} \). Thus, we have to use heuristics to prune out unlikely fragmentations and then generate fragmentations within the 1804 sentences.

We base the pruning on fragmentations that actually generated by human annotators as explained below.

![Histogram of number of tokens for 132 human-annotated fragments](image)

**Figure 5.3:** Histogram of number of tokens for 132 human-annotated fragments. This diagram uses the Quantity column from the table shown on the top right-hand side. The first column in the table represents the left point side of each interval. The second column is the interval width, and the third column (Quantity) represents the number of sentences that fall into each interval.

We consider the fragmentations within a corpus consisting of 10,000 sample sentences that were annotated by the human experts. The number of fragments within
these fragmentations is 132. Figure 5.3 shows the distribution of the number of tokens for these 132 fragments. As can be seen, the number of tokens in most of these fragments ranges between 4 and 30. Thus to avoid generating unlikely fragmentations, we add another constraint indicating that the number of tokens in all the fragments within an informative fragmentation must be between 4 and 30.

Figure 5.4: Histogram of number of informative fragmentations for all sentences. Table on the right corner lists the quantities being used in the histogram and consists of three columns. The first column represents the left point side of each interval. The second column is the interval width, and the third column represents the number of sentences that fall into each interval.

The distribution of possible informative fragmentations for the 1804 sentences after applying the above constraint is shown in Figure 5.4. As can be seen, most of the sentences have only one informative fragmentation, which is comprised of the whole sentence.

Second, the optimal informative fragmentation within each of the 1804 sentences
was chosen. As previously discussed, there is a joint entropy value associated with each optimal fragmentation. Figure 5.5 shows the distribution of joint entropy values for the selected fragmentations. The joint entropy value is inversely proportional to the certainty level of fragment classifications.

We next describe the properties of the optimal fragmentations generated by our automated fragmentation algorithm in more detail. We then measure in Section 5.4.2 the similarity between fragmentations resulting from our proposed algorithm and the fragmentations assigned by human experts.

### Fragmentation by the Automated Fragmentation Algorithm

Unlike human experts that only fragmented 40 sentences into two or more fragments, our algorithm fragmented 501 sentences into two or more fragments, while keeping
the remaining 1303 sentences unfragmented. Among the 501 sentences that were fragmented by our proposed algorithm, 31 are among those 40 sentences that were also fragmented by human experts, while 470 sentences are fragmented only by our algorithm but not by human experts.

The distribution of the number of fragments per sentence is shown in Figure 5.6.

![Bar chart showing the distribution of number of fragments per sentence.](image)

**Figure 5.6:** *The distribution of number of fragments for sentences fragmented by the fragmentation algorithm.*

In Section 5.4.2, we discuss the results of measuring similarity between the fragmentations generated by our algorithm and those assigned by human experts.

### 5.4.2 Edit-Distance Results

The edit-distance values shown in Figure 5.7(a) indicate the distances between fragmentations obtained using our algorithm to those from the ground truth. Among
1804 sentences, 1299 were perfectly fragmented by the algorithm, that is, the fragmentations were the same as those in the human annotated set. The percentage chart is plotted in Figure 5.7(b).

As most of the results from our algorithm, and most of the human-annotated fragmentations, both leave the majority of the sentences whole (unfragmented), we focus now on the 40 sentences in the set that were fragmented into two or more fragments by human experts.

The edit-distance values for these 40 sentences are shown in Figure 5.8(a). As illustrated in Figure 5.8(a), the majority of the edit-distances have a value of one. Here, a value of one indicates two different cases. One case is when our proposed algorithm fragments sentences one token before or after the correct fragmentation point. In this case, the value of the edit-distances is the cost for the separator-move operator (this case is shown in black in Figure 5.8(a)). The other case is when our fragmentation algorithm leaves the sentences unfragmented while the human annotators fragmented the sentences into two fragments. The value of the edit-distances in the second case is the cost of the separator-insertion operator (shown in grey in Figure 5.8(a)).

In Section 5.5, we analyze the results shown in Figure 5.8. We also discuss several issues that may lead to the mis-fragmentations shown in Figure 5.8(b).
(a) The distribution of Edit-Distance values calculated between the fragmentations generated by our proposed algorithm and the ground truth for 1804 sentences.

(b) The percentage of the edit-distance values shown in (a).

Figure 5.7: Edit-distance values for 1804 sentences.
(a) The distribution of Edit-Distance values calculated between the fragmentations generated by our proposed algorithm and the ground truth for 40 sentences. The grey part represents the case when our fragmentation algorithm leaves the sentences unfragmented while the human annotators fragmented the sentences into two fragments.

(b) The percentage of the edit-distance values shown in (a).

Figure 5.8: Edit-distance values for the 40 sentences that were also fragmented by human annotators.
5.4.3 Randomly Generated Fragmentations

In this section, we calculate the edit-distance between the randomly generated fragmentations discussed in Section 5.3.2 and human fragmentations. The edit-distance values are shown in Figure 5.9. Visually comparing the two graphs shown in Figures 5.9 and 5.7(a), the edit-distance values in Figure 5.9 are lower than the ones in Figure 5.7(a). The random fragmentation algorithm is told not to fragment sentences 97% of the times. As almost all the sentences are not fragmented by the experts and the same ratio remains unfragmented at random, this tremendous agreement between the random and the human fragmentation when there is no fragmentation is expected.

![Figure 5.9: Edit-distance values between randomly generated fragmentations and ground truth for 1804 fragmentations.](image)

We therefore focus on the edit-distance values for only the 40 sentences that were also fragmented by the human annotators. The results are shown in Figure 5.10. As can be seen from this figure, the random algorithm did not fragment 37 of the
sentences that were fragmented by the human experts while the remaining 3 sentences were fragmented incorrectly.

For these 40 sentences, we compare these edit-distance values with the edit-distance values calculated between the resultant fragmentations from our algorithm and the human fragmentations to which we refer as ground truth. The mean squared error with respect to the human-generated fragmentations is used to calculate the error related to our proposed algorithm and to the algorithm that randomly generates fragmentations:
\[ MSE = \frac{1}{k} \sum_{i=1}^{k} e_i^2, \]  

where \( k \) is the number of sentences and \( e_i \) is the edit-distance value for the \( i \)th fragmentation.

As discussed in Section 5.4.2, an edit-distance value of one indicates two different cases: one case is when the proposed algorithm fragments sentences one token before or after the correct fragmentation point; the other case is when the proposed algorithm does not fragment sentences while the human experts do. When a sentence was fragmented by a human annotator, intuitively we view it as better when the algorithm fragments the sentence at the wrong position than when it does not fragment the sentence at all. To distinguish between the two cases, we modify the formulation presented in Equation 5.2 as follows:

\[ MSE' = \frac{1}{k} \sum_{i=1}^{k} (e_i + P_i)^2, \]  

where \( P_i \) is the penalty value assigned to those sentences that are not fragmented by our algorithm while they are fragmented by the human experts. For the sentences that are fragmented by our algorithm, this penalty value is equal to zero. All the edit-distance values except zero and one (only the ones that are associated with a case that fragmentation is performed wrongly) that are shown in Figures 5.8(a) and 5.10 are related to the case when our algorithm fragments the sentence at the wrong position. Hence, to distinguish between the case when the algorithm fragments the sentence wrongly and the case when it does not fragment the sentence at all, the penalty value must be more than the maximum edit-distance value shown in Figures 5.8(a) and
5.10. Here this penalty value is therefore set to 5.

Our proposed algorithm did not fragment 9 of the sentences that were fragmented by the human experts. Thus, the $MSE'$ value for our proposed algorithm is calculated as follows:

$$MSE'_1 = \frac{1}{40}(11(1^2) + 9(1^2 + 5^2) + 11(2^2) + 2(3^2) + 2(4^2)) = 8.47.$$  

Furthermore, out of the entire 40 sentences that were fragmented by the human experts, the random fragmentation algorithm did not fragment 37. Hence the error for the algorithm that randomly generates fragmentations is calculated as:

$$MSE'_2 = \frac{1}{40}(2(1^2) + 37(1^2 + 5^2) + 1(2^2)) = 24.2.$$  

By comparing $MSE'_1$ and $MSE'_2$, we see that our algorithm clearly outperforms the randomly generated fragmentation algorithm.

### 5.5 Results Analysis

As discussed in Section 5.4.2, among 1804 sentences, 1299 were perfectly fragmented, that is, the edit-distance values between the fragmentations of these sentences and the ground truth are zero. For the other 505 sentences, we discuss possible problems that may have had an impact on the performance of the proposed fragmentation algorithm, and its evaluation. Each problem is discussed in detail in the following sections.

#### 5.5.1 Classification Performance

Classification results can directly alter the fragmentation results. The reason is that a sentence break occurs at the point where there is a change in the class label along
one of the Focus or Evidence dimensions.

We have the human class-assignments only for the fragments generated by our algorithm that exactly match the human-generated fragments. We thus do not have any ground-truth, human-assigned, classification for other fragments generated by our algorithm. Without having human-assigned classes, we are not able to directly assess the classification performance.

To have an estimate of how well our classifier performs, we can generate the same fragments as the ones generated by human experts and classify them using our algorithm and then measure the classification performance.

In this section, we first measure the classification performance for all 1844 fragments in Sent-FE-INFO. The classification performance is measured in terms of Precision, Recall and F-measure. Precision and Recall of a classifier for a category $c$ are defined as follows:

$$\text{Precision} = \frac{TP_c}{TP_c + FP_c},$$

$$\text{Recall} = \frac{TP_c}{TP_c + FN_c};$$

where $TP_c$ indicates the number of true positives, that is, the number of fragments correctly classified into category $c$ and $FN_c$ indicating the false negatives, that is, the number of fragments incorrectly rejected from $c$. Additionally, as we previously discussed in Section 4.1, the F-measure is a popular combination of Precision and Recall, defined as follows:

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$  

Tables 5.2 and 5.3 show the performance of the Focus and Evidence classification for the 1844 fragments in Sent-FE-INFO.
Table 5.2: The performance of the Focus classification for 1804 sentences (1844 fragments). S denotes Scientific, G denotes Generic, and M denotes Methodology.

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0.96</td>
<td>0.98</td>
<td>0.97</td>
</tr>
<tr>
<td>G</td>
<td>0.96</td>
<td>0.57</td>
<td>0.71</td>
</tr>
<tr>
<td>M</td>
<td>0.96</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>Average ($\frac{S+G+M\text{ scores}}{3}$)</td>
<td>0.96</td>
<td>0.83</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 5.3: The performance of the Evidence classification for 1804 sentences (1844 fragments). E0 denotes No evidence, E1 denotes Claim of evidence without verifying information, and E2 denotes Explicit citation, and E3 denotes Explicit evidence.

<table>
<thead>
<tr>
<th>Category</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>0.76</td>
<td>0.96</td>
<td>0.84</td>
</tr>
<tr>
<td>E1</td>
<td>0.90</td>
<td>0.57</td>
<td>0.69</td>
</tr>
<tr>
<td>E2</td>
<td>0.96</td>
<td>0.83</td>
<td>0.89</td>
</tr>
<tr>
<td>E3</td>
<td>0.97</td>
<td>0.91</td>
<td>0.93</td>
</tr>
<tr>
<td>Average ($\frac{E0+E1+E2+E3\text{ scores}}{4}$)</td>
<td>0.89</td>
<td>0.81</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Now, we consider only fragments within the 40 sentences that were also fragmented by human experts. As noted in Section 5.3, human experts fragmented all these 40 sentences into two fragments. The class labels for all the 80 generated fragments are then predicted by the classifiers presented by Pan et al. [33, 49].

The classification decisions along the Focus and Evidence dimensions for these 80 fragments are shown in Table 5.4 and 5.5 respectively. Each row represents a true class label, and each column represents a predicted class label by our algorithm. Out of the entire 80 fragments, there are only 8.5 mis-classified fragments. Moreover,
as shown in the confusion matrix in Table 5.5, there are mainly two kinds of misclassifications along the Evidence dimension; one relates to the mis-classification of category E3 as E0 and the other is the mis-classification of E1 as other categories. In this case, there are only 14 mis-classified fragments.

We have to examine whether the mis-classified instances indeed correspond to mis-fragmented sentences. As shown in Figure 5.8(a), among 40 sentences, only 5 of them were perfectly fragmented. The mis-classifications are related to 20 sentences that were also mis-fragmented by our algorithm. For the remaining 15 sentences, mis-classification is less likely to be the cause of mis-fragmentation.

Table 5.4: The confusion matrix of the Focus classification for 80 fragments. Each row represents a true class label, while each column represents a predicted class label.

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>G</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>68</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>G</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>0</td>
<td>2.5</td>
</tr>
</tbody>
</table>

Table 5.5: The confusion matrix of the Evidence classification for 80 fragments.

<table>
<thead>
<tr>
<th></th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>31</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>E2</td>
<td>4</td>
<td>0</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>E3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
</tbody>
</table>

The classification decisions for the fragments within the 470 sentences that were fragmented by our algorithm but not by the human experts were also studied. The
results are shown in Table 5.6 and 5.7.

Table 5.6: *The confusion matrix of the Focus classification for 470 fragments.*

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>G</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>315.5</td>
<td>0</td>
<td>6.5</td>
</tr>
<tr>
<td>G</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>M</td>
<td>8.5</td>
<td>0</td>
<td>126.5</td>
</tr>
</tbody>
</table>

Table 5.7: *The confusion matrix of the Evidence classification for 470 fragments.*

<table>
<thead>
<tr>
<th></th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>E1</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>E2</td>
<td>49</td>
<td>0</td>
<td>166</td>
<td>0</td>
</tr>
<tr>
<td>E3</td>
<td>21</td>
<td>1</td>
<td>3</td>
<td>199</td>
</tr>
</tbody>
</table>

As can be seen from Table 5.6, there are a few mis-classification along the Focus dimension. However, from the confusion matrix shown in Table 5.7, there is more mis-classification along the Evidence dimension. Out of the entire 470 sentences, these mis-classifications are related only to 90. Thus, we can conclude that only for a few sentences, the mis-fragmentation was a result of mis-classification.

### 5.5.2 Additional Knowledge of Human Expert

By examining the fragments generated by human experts, we can conclude that most of the fragments generated by human experts are meaningful phrases or combination of phrases. As an example, consider the following sentence consisting of two fragments:
Furthermore, we show that the increased somal \( \text{[Ca}^{2+}]_i \) \(^{**E3}\) and decreased cell survival following proximal transactions are not due to less frequent or slower plasmalemmal sealing or Ca\(^{2+}\) entry through plasmalemmal Na\(^{+}\) and Ca\(^{2+}\) channels. \(^{**E0}\)

Informative terms for the Focus and Evidence dimensions are shown in bold. As can be seen, the sentence is fragmented into two fragments. In the first fragment, the phrase “We show” indicates that the evidence has been explicitly provided somewhere within the paper; therefore, the Evidence level is \( E3 \), that is, Explicit evidence. In the second fragment, there is no indication of evidence, and accordingly, the Evidence level is \( E0 \). Additionally, based on the informative terms present in both fragments, the Focus of both fragments is Scientific. Moreover, we can see that each fragment consists of a noun phrase and a verb phrase.

Human experts used an implicit rule in order to break the sentence into meaningful fragments. The implicit rule used is that “a sentence break never occurs in the middle of a phrase”. However, fragments generated by our proposed algorithm do not impose such a rule. For example, the following fragments result from applying our algorithm to the same sentence:

Furthermore, we show that the increased somal \( \text{[Ca}^{2+}]_i \) and decreased cell survival following proximal transactions are not due to less frequent or slower plasmalemmal sealing or Ca\(^{2+}\) entry through plasmalemmal
As can be seen, as in the fragmentation generated by human experts, the fragmentation resulting from our algorithm is motivated by the changes in Evidence from Explicit evidence \( (E3) \) to No evidence \( (E0) \). However, for the first fragment, a sentence break occurs in the middle of a phrase “following proximal transactions”. By imposing an additional rule on the generated fragments, the same fragmentation as the one generated by human experts may be generated automatically. However, the fragmentation generated by our current method is not incorrect for the end-goal of fragment classification, since the resulting fragments still both belong to the correct classes, and still capture the change in evidence as intended.

As another example, consider the following sentence:

Interestingly, TGF-beta signaling can directly interact with WNT signaling by a complex between beta-catenin T-cell factor / Leukemia enhancer factor and Smad4 on the Xtwn promoter \( (53) \). **SE0

The reference “(53)” indicates that there is an Explicit citation, that is, the Evidence level is \( E2 \). The following fragments result from applying our algorithm to the same sentence:

Interestingly, TGF-beta signaling can directly interact with WNT signaling by a complex between beta-catenin T-cell factor / Leukemia enhancer factor and Smad4 on the Xtwn promoter \( (53) \). **SE2

76
As can be seen, without taking into account the human experts rule, our algorithm fragmented the sentence into two fragments. This fragmentation is not incorrect, in the sense that the first part of the sentence indeed does not carry evidence. Clearly, an algorithm, that identifies all the words following the term “complex” as relating to the complex and as supported by citation (53) would avoid the unnecessary fragmentation. Additional examples are provided in Appendix A.

5.6 Summary

In this chapter, we presented an experimental framework for applying our automated fragmentation algorithm to 1804 sentences along the Focus and Evidence dimensions. Based on the results presented in Section 5.4.2, among 1804 sentences, 1299 were perfectly fragmented, that is their resulting fragmentations are the same as the ground truth fragmentations.

We took a closer look at the fragmentations of two types of sentences. First, the ones that were also fragmented by the human experts (40 sentences). Second, the ones that were fragmented by our proposed algorithm but not by the human experts (470 sentences).

In both types of sentences, only for a few, the mis-fragmentation was a result of mis-classification. For the remaining, the major issue appears to be that the human experts used an implicit rule regarding not breaking a sentence in the middle of a phrase, while our proposed algorithm does not consider that rule. However, for all the incorrect fragmentations, the classification results for the fragments within them are still correct.
Our ultimate goal is to classify sentence fragments generated by our algorithm using the multi-dimensional classifier proposed by Pan et al. [33, 49] and make further use of them in follow-up applications. For applications that aim to address certain types of information needs, as long as the sentence fragments are classified correctly, fragmenting a sentence a few terms to the left or to the right is acceptable, because the same information can still be captured. We believe that by proposing an evaluation measure that better suits the specific target application, the fragmentation performance will be better measured and the score may be higher according to the new measure.
Chapter 6

Conclusion and Future Work

There has recently been great interest in the development of text mining applications to make more effective use of the knowledge contained in biomedical publications. However, most of the research in biomedical text mining focuses on extracting certain information from scientific text. We note that different researchers have different information needs. Pan et al. [33, 49] proposed a method aiming to categorize text in order to address each user specific information needs. They aim to automatically categorize biomedical sentence fragments into predefined categories grouped into the five dimensions: Focus, Evidence, Certainty, Polarity, and Trend. In the multi-dimensional classification scheme proposed by Pan et al. [33, 49], the basic classification unit is defined as a sentence fragment. Thus, automatically breaking sentences into fragments is an important first step in fragment classification.

The work presented here examined the feasibility of the automatic fragmentation of sentences along only the Focus and Evidence dimensions. In our proposed approach, for each sentence, we first generated all the possible fragmentations that meet the criteria discussed in Sections 3.1.2 and 5.4.1. Then, an entropy-based score
function was defined in order to find the optimal fragmentation within that sentence. We assigned a score to each generated fragmentation based on the certainty reflected in the probability distribution over all the possible class labels given each fragment within that fragmentation. The high score value, or low entropy value, corresponds to a high certainty inherent in each probability distribution, \( Pr(C|F) \), where \( C \) is a random variable denoting the possible category label and \( F \) is a sentence fragment. Finally, for each sentence, among all the generated fragmentations, the highest scoring fragmentation was selected as an optimal fragmentation.

We also proposed a new evaluation measure, namely, edit-distance with separator-move operation, to assess the distance between the fragmentations resulting from applying our algorithm and the human-annotated fragmentations. For our fragmentation task, our new evaluation measure addresses the limitation of standard evaluation measures such as: Precision, Recall, and the edit-distance with only the two operations insertion and deletion. In the proposed approach, the permitted edit operations are separator-move, separator-insertion, and separator-deletion. Each edit-distance value represented the total cost of the edit operations that can be used to transform one fragmentation into the other.

The results have shown that only a few mis-fragmentations were caused by misclassification of fragments by Pan’s classifier [33, 49]. The main reason for mis-fragmentation appears to be that the human experts break sentences at phrase boundaries, while our algorithm may break a sentence in the middle of a phrase. We noted that since the class labels assigned to the fragments generated by our algorithm were still typically correct, these mis-fragmentations should probably not be considered as incorrect fragmentations.
There are several possible extensions to this work. As one possible direction, we plan to extend the work to automatically fragment sentences along the other three dimensions: *Polarity*, *Certainty*, and *Trend*. We then have to integrate the fragmentations results along all the five dimensions to get the final fragmentation result.

Moreover, as noted in Section 5.4, as long as the generated fragments are classified correctly, fragmenting sentences a few terms to the left or to the right from the position where a human-annotator may have fragmented it is not critical. The reason is that the target applications for our proposed algorithm are concerned with the type of knowledge contained within the fragments rather than the exact fragment boundaries. Hence, a further extension to this work is to propose a new evaluation measure that can accurately capture the performance of our proposed fragmentation algorithm based on the target application.

We believe that our proposed fragmentation algorithm can be used in other domains such as sentiment analysis. As described in Section 2.5.4, sentiment analysis attempts to classify text into two categories: *positive* or *negative*, and can be used to automatically predict the sentiments of document, sentence, or phrase (fragment). Our fragmentation algorithm can be used as a step in phrase-level sentiment classification, in order to first break sentences into phrases. A sentence break must occur at a point where there is a change in the sentiment of a sentence. The performance of sentiment classification depends on the quality of the phrases resulting from our fragmentation algorithm.
Bibliography


Appendix A

Examples of Different Fragmentations

In this section, for 5 sentences, we show human annotators (HA) fragmentations and the fragmentations resulting from our proposed algorithm (AF). The informative terms along the Focus and Evidence dimensions are shown in bold.

1. **HA:** Given that the majority of colorectal cancers involve activation of the beta-catenin signaling pathway, and given that multiple mutations lead to this activation, **SE0** there is a clear need for drugs that attenuate the nuclear functions of beta-catenin (15). **GE2**

   **AF:** Given that the majority of colorectal cancers involve activation of the beta-catenin signaling pathway, and given that multiple mutations lead to this activation, there is **SE0**
a clear need for drugs that attenuate the nuclear functions of beta-catenin (15). **GE2

2. **HA: Another potential factor in promotion of apoptosis, inducible NO synthase (37), **SE2 is limited in distribution to perivascular infiltrates at the peak of inflammation and is unlikely to contribute to widespread neuronal loss. **SE0

AF: Another potential factor in promotion of apoptosis, **SE0 inducible NO synthase (37), is limited in distribution to perivascular infiltrates at the peak of inflammation **SE2 and is unlikely to contribute to widespread neuronal loss. **SE0

3. **HA: In sharks, this protein has a polypeptide backbone of 26.8 kDa (20) **SE2 which contains within its primary sequence three potential sites for N-linked glycosylation in a putative extracellularly disposed hydrophilic domain. **SE0

AF: In sharks, this protein has a polypeptide backbone of 26.8 kDa (20) which contains within its primary sequence three potential sites for N-linked glycosylation in a putative extracellularly disposed hydrophilic domain. **SE0
4. **HA:** Since the **three**-amino acid insertion does **not** alter the way in which the homeodomain **binds DNA (36) **SE2** the insertion **may** have originated as a benign **mutation** that only later acquired the **function** of mediating interactions with other homeodomain **proteins**. **SE0**

**AF:** Since the **three**-amino acid insertion does **not** alter the way in which the homeodomain **SE0** binds DNA (36) the insertion **may** have originated as a benign **mutation** that only later acquired the **function** of mediating interactions with other homeodomain **proteins**. **SE2**

5. **HA:** GSK-3beta **is** a negative regulator of **Wnt signalling (4) **SE2** which **is** inactivated by **phosphorylation mediated by Wnt signals or ILK. **SE0**

**AF:** GSK-3beta **is** a negative regulator of **Wnt **SE0** signalling (4) which **is** inactivated by **SE2** phosphorylation mediated by Wnt signals or ILK **SE0**
Appendix B

Program Source Codes

B.1 Text Preprocessing system for Automated Biomedical Text Fragmentation

Server: redtape.cs.queensu.ca

B.1.1 To represent the human annotators fragments

Home Directory: /fs/hs/projects/AutoFrag/textProcess

COMPILING
To compile the source, GNU make and GNU g++ (version 3.4.x or higher recommended) are required. Simply typing the command “make” or “gnumake” to compile the .cpp source and generate the executable file: textProcessor.

Location: /fs/hs/projects/AutoFrag/textProcess/makefile
COMMAND

./textProcessor -d [...] -matlab -represent [...] -n [...] -output outputFile

COMMAND OPTIONS

• **-d**: Choose a dimension along which the text is processed:
  
  – F: Focus
  
  – E: Evidence

• **-matlab**: Generate the text representation for Matlab, i.e., no feature names as column headers, and the class labels are numerical.

• **-represent**: Whether the pre-processing is done on the pre-labeled fragments or non-pre-labeled:
  
  – p: Pre-labeled fragments
  
  – n: Non-pre-labeled fragments

• **-n**: Set the number of informative terms along a specific dimension to be included in the text representation.

• **-output**: Specify the name of the text representation file.

INPUT FILE

The input files are located in ./textProcess/Input

OUTPUT FILE
The output files will be stored in ./textProcess/TrainData and further will be used to train a classifier.

**COMMAND EXAMPLES**

./textProcessor -d F -matlab -represent p -n 600 -output TrainData_F.csv
./textProcessor -d E -matlab -represent p -n 200 -output TrainData_E.csv

**B.1.2 To break input sentences into their possible informative fragmentations**

**Home Directory**: /fs/hs/projects/AutoFrag/textProcess

**COMPILING**

To compile the source, GNU make and GNU g++ (version 3.4.x or higher recommended) are required. Simply typing the command “make” or “gnumake” to compile the .cpp source and generate the executable file: textProcessor.

**Location**: /fs/hs/projects/AutoFrag/textProcess/makefile

**COMMAND**

./textProcessor -breakFrag -matlab -represent [...]  

**COMMAND OPTIONS**

- **-breakFrag**: The task is to break sentences into their informative fragmentations
- **matlab**: Generate the text representation for Matlab, i.e., no feature names as column headers, and the class labels are numerical.

- **represent**: Whether the pre-processing is done on the pre-labeled fragments or non-pre-labeled:
  
  - p: Pre-labeled fragments
  
  - n: Non-pre-labeled fragments

**INPUT FILE**
The input files are located in ./textProcess/Input

**OUTPUT FILE**
The output files will be stored in ./textProcess/TestData and further will be used in order to select the optimal fragmentations within sentences.

**B.1.3 To train the classifier**

**Home Directory**: /fs/hs/projects/AutoFrag/Machine.Learning

Run MEInit.m

**INPUTS**

- The file TrainData-F.csv located in /fs/hs/projects/AutoFrag/textProcess/TrainData

- The file TrainData-E.csv located in /fs/hs/projects/AutoFrag/textProcess/TrainData
OUTPUT

The model parameters of the classifier trained on the human annotators fragmentations.

B.1.4 To select optimal fragmentations within sentences

Home Directory: /fs/hs/projects/AutoFrag/Machine_Learning

Run compareJEs.m

INPUTS

- The file TestData_F.csv located in /fs/hs/projects/AutoFrag/textProcess/TestData
- The file TestData_E.csv located in /fs/hs/projects/AutoFrag/textProcess/TestData

OUTPUTS

- A structure named Sent that holds the following for each sentence:
  - The ID of a sentence (SentID).
  - The break points indexes of the optimal fragmentation within a sentence (BestFragmentation).
- A structure named Fragmentation that holds the following for each sentence:
  - The joint probability value (probs).
  - The break points indexes (Indexes).
  - The joint entropy value (JointEntropy).
Focus labels (\textit{Labels1}).

Evidence Labels (\textit{Labels2}).

- **MJE**: An array containing the minimum Joint Entropy values.

- **\textit{nFrags}**: An array containing the number of possible informative Fragmentations for each Sentence.

- **\textit{R.txt}**: Each line in this text file represents the fragmentation points within each sentence resulting from our fragmentation algorithm.

### B.1.5 To randomly fragment sentences

**Home Directory**: /fs/hs/projects/AutoFrag/Machine.Learning

**Run** Random.m

**OUTPUT**

\textit{Random.txt}: The file that contains the randomly generated fragmentation points for sentences that were also fragmented by human annotators.

### B.1.6 To calculate the edit-distance between ground truth fragmentations and the fragmentations resulting from our algorithm

**Home Directory**: /fs/hs/projects/AutoFrag/Edit-DistanceCalc
COMMAND
./Evaluation

INPUT FILES

• *GS.txt*: The file that contains the ground truth fragmentation points and is located at ./textProcess/tempFile

• *R.txt*: The file that contains the fragmentation points resulting from our algorithm and is located at ./machinlearning

OUTPUT

*EditDistance.txt*: Each line of this file represents the edit-distance value between a ground truth fragmentation and a fragmentation resulting from our algorithm.

**B.1.7 To calculate the edit-distance between ground truth fragmentations and the randomly generated fragmentations**

**Home Directory**: /fs/hs/projects/AutoFrag/Edit-DistanceCalc

COMMAND

./Evaluation

INPUT FILES

• *GS.txt*: The file that contains the ground truth fragmentation points and is located at ./textProcess/tempFile
• *Random.txt*: The file that contains the randomly generated fragmentation points and is located at `./machinlearning`

**OUTPUT**

*EditDistanceRandom.txt*: Each line of this file represents the edit-distance value between a ground truth fragmentation and a randomly generated fragmentation.