Sentence unit detection using lexical information for automatic speech recognition transcripts

Ho, Thi Nga

2018

Ho, T. N. (2018). Sentence unit detection using lexical information for automatic speech recognition transcripts. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/103496>

<https://doi.org/10.32657/10220/47326>

**SENTENCE UNIT DETECTION USING LEXICAL
INFORMATION FOR AUTOMATIC SPEECH
RECOGNITION TRANSCRIPTS**

HO THI NGA

School of Computer Science and Engineering

**A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering**

2018

Abstract

This thesis studies Sentence Unit Detection (SUD) that uses lexical information for Automatic Speech Recognition (ASR) transcripts. An SUD system detects the presence or absence of a sentence unit boundary at every word boundaries of an unpunctuated text document. A transcript produced by an ASR system from a given speech audio input is an unpunctuated text document. Hence, SUD for ASR transcripts will increase the readability and enable the usability of ASR transcripts in downstream text processing applications, such as machine translation and option mining, which require sentence unit boundary information in their input. As usually being a post-processing application for ASR transcripts, an SUD system may benefit from two major information sources, i.e., the prosodic information obtained from the audio input and the lexical information obtained from the ASR transcripts, for its model learning. In this thesis, we chose to use only lexical information because it has shown to be essential for SUD research as compared to the other. Moreover, using lexical information takes an advantage of the massive available lexical data sources from the Internet. Additionally, SUD models that use only lexical information are usually much less complex as compared to the models that employ multiple information sources.

To improve an SUD system performance, one possible direction is to improve the quality of input features. Previous studies usually paid more attention on finding new input features to enrich and enlarge the existing feature space rather than refining it for a better quality. This tendency might lead to feature redundancy and unnecessarily increase the model complexity. This thesis addresses the lack in research by studying approaches that fine tune existing lexical features to SUD systems. Specifically, there are two proposed approaches: The first approach focuses on optimizing the existing features to reduces the input feature space size. The second study focuses on evaluating

the usefulness of many variants of the same feature type to the performance of a SUD system.

The first study proposes a feature selection method to optimize the distant-bigram features as the input to a CRF based SUD system. The target is to reduce the system’s model complexity but maintaining a comparable system performance. The study uses Pointwise Mutual Information (PMI) as a feature selection method, to select only informative features for model training, to reduce training cost and model complexity. PMI is used to measure the correlation of the input features to the presence of the sentence unit boundary at the hypothesized word boundary. The obtained PMI values are then used as selection criteria, i.e. the higher the PMI value obtained for a feature, the higher the chance that the feature is selected for model training. The proposed method significantly reduces the size of the input feature space by 44.87% relatively while still maintaining comparable performance to the original model which has no feature selection. CRF is known to be a shallow learning technique which usually heavily relies on feature extraction engineering to achieve a good prediction performance. Thus it is reasonable to get a better result in term of model complexity and performance from fine tuning the input feature space.

However, the recent increase in popularity of using deep learning in SUD research has shown to be less dependent on feature extraction engineering. Thus, it is questionable if the fine tuning on the existing input features will still be helpful. To answer this, our second study proposes a framework to evaluate the effectiveness of using different variants of the same feature type, i.e., word embedding, as input to train a deep learning based SUD model. Specifically, two word embedding variants, i.e., word-based and subword-based skip-gram, are trained using either a small but in-domain dataset or a large but out-of-domain dataset, resulting four variants of word embedding models, i.e. in-domain word-based, out-of-domain word-based, in-domain subword-based and out-of-domain subword-based. Subsequently, each of the four word embedding models is used to extract embedding feature as an input to a deep neural network with biLSTM and fully connected layers to train separate SUD models. Our study reveals that the in-domain subword-based embedding gives the best performance in all testing conditions. On the other hand, the word-based embedding model is recommended to use in the case that there is no in-domain dataset available for word embedding model training.

Acknowledgement

I want to express my sincerest thanks to my supervisor, Dr.Chng Eng Siong, for his invaluable support and guidance.

I also thank all my colleagues in Temasek Laboratories @NTU and the SCSE Speech Recognition team, who are not only co-workers but also friends. Especially, I would like to thanks Dr. Chong Tze Yuang, Ms. Vu Thi Ly, Mr. Xu Chenglin, Ms. Aditi Ghosh and Mr. Can Duy Cat for the help and support they have given to me.

Last but not least, thanks to my family for their unconditional love, support, and encouragement. Without them, my research life would be much harder. I also want to give a special thanks to my husband and my little girl for always being beside me, whatever and whenever, the situation.

Contents

List of Publications	vii
List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Motivation	1
1.1.1 ASR Transcripts	1
1.1.2 Sentence Unit Detection	2
1.2 Objective and Thesis Scope	6
1.3 Contributions	7
1.4 Thesis organization	8
2 An Overview of Sentence Unit Detection for ASR Transcripts	9
2.1 Sentence Unit Detection	9
2.2 Features Extraction	10
2.2.1 Lexical Features	10
2.2.2 Prosodic Features	15
2.3 Modeling Approaches	19
2.3.1 Feature Incorporation Strategy	19
2.3.2 Machine Learning Techniques	21
2.3.2.1 Shallow Learning	22
2.3.2.2 Deep Learning	25
2.3.3 Semi-supervised Approaches	27
2.4 Conclusion	28

3	Optimizing Input Feature Space Using Pointwise Mutual Information	30
3.1	Introduction	31
3.2	Feature Extraction	32
3.3	Conditional Random Fields For SUD Modeling	32
3.4	Proposed framework	35
3.4.1	Pointwise Mutual Information	36
3.4.2	PMI-based Feature Selection	37
3.4.2.1	Feature Selection For Multiple Distant-bigram Features	38
3.5	Experiments	38
3.5.1	Data Preparation	39
3.5.2	Experimental Setup	40
3.5.3	Parameter Tuning	41
3.5.4	Applying Feature Selection	43
3.6	Conclusion	43
4	Investigating Word Embeddings for Deep Learning based SUD	45
4.1	Proposed Framework	46
4.2	Word Embedding Extraction	47
4.2.1	Word Embedding Types	47
4.2.2	Extraction Strategy	49
4.3	SUD Model Learning	49
4.4	Experiments	53
4.4.1	Data and Evaluation Metrics	53
4.4.2	Experimental Setup	55
4.4.3	Evaluation Results	56
4.5	Discussion	58
4.5.1	Related Work	58
4.5.2	Another generalization test on a written dataset	59
4.6	Conclusion	60

5	Conclusions	62
5.1	Contributions	62
5.1.1	Pointwise Mutual Information for feature selection	62
5.1.2	Study of word embedding types	63
5.2	Future Work	63
	References	66

List of Publications

- (i) **Thi-Nga, Ho** Tze-Yuang, Chong, Van-Hai, Do, Van-Tung Pham and Eng-Siong, Chng. Improving Efficiency of Sentence Boundary Detection by Feature Selection. In Asian Conference on Intelligent Information and Database Systems (pp. 594-603). Springer, Berlin, Heidelberg, March 2016.
- (ii) **Thi-Nga, Ho**, Duy-Cat, Can and Eng-Siong, Chng. An Investigation of Word Embeddings with Deep Bidirectional LSTM for Sentence Unit Detection in Automatic Speech Transcription. *Accepted for* presentation in the IALP 2018 - the International Conference on Asian Language Processing. Bandung, Indonesia, August 2018.
- (iii) Duy-Cat, Can, **Thi-Nga, Ho** and Eng-Siong, Chng. A Hybrid Deep Learning Architecture for Sentence Unit Detection. *Accepted for* presentation in the IALP 2018 - the International Conference on Asian Language Processing. Bandung, Indonesia, August 2018.

List of Figures

1.1	Enriching ASR transcripts by (1) removing disfluencies and non-speech words, (2) sentence unit detection and (3) capitalization (including name entity detection).	3
1.2	An overview of the SUD task as a post-processing step after the ASR system produces a transcript from the audio input.	4
1.3	A common SUD framework.	5
1.4	An example of a SUD as a sequential tagging task.	6
2.1	An example of common feature sets used for LM[1].	11
2.2	An example of n-gram used to predict SU labels.	11
2.3	Examples of CBOW and skip-gram representations.	13
2.4	An example of an input sequence with its POS and Chunk tags.	14
2.5	An aligned speech signal and word level transcript for a spoken utterance. Clear long pauses are observed at the sentence ending word boundary, i.e., “day” and lunch.	16
2.6	Examples of F0 features: F0 slope, F0 Continuity, F0 reset and F0 range.	17
2.7	Frame-based energy progression of a speech signal with energy decay (red) in [dB/s] within potential sentence boundaries (green) [2].	18
2.8	Group of modelling approaches based on feature type used	20
2.9	A top-down tree map for modelling approaches in SUD research.	22
3.1	The baseline SUD system work-flow.	31
3.2	An example of multiple distant-bigram features for the word boundary at $w_6 = \text{“day”}$ in the sequence. Here $-6 \leq k \leq +6$ and $k \neq 0$.	33

3.3	An example of building feature functions for the word boundary being hypothesized at w_6 ="day" using its twelve distant-bigram features. . . .	34
3.4	An example of parameter learning.	35
3.5	The proposed framework with feature selection applying after feature extraction and before modeling.	36
3.6	An example of PMI scores obtained for distant-bigram features of the word boundary being hypothesized at w_6 ="day". The PMI score corresponding to each feature is obtained using Equation 3.2.	37
3.7	An example of PMI feature selection for word boundary being hypothesized at w_6 ="day". Here, features are distant-bigram with $-6 \leq k \leq +6$ and $k \neq 0$. Only features that have PMI scores higher than 0.30 are selected, i.e. the scores highlighted in bold	38
3.8	An example of PMI feature selection applied to multiple distant-bigram features of the entire input sequence. Here, features are distant-bigram from $k = -6$ to $k = +6$. Only features that have PMI scores higher than the corresponding selection threshold p_k are selected. The p_k varies at different values of k	39
3.9	F1-scores (y-axis) obtained when context size n was varied from 2 to 12.	41
3.10	Feature selection at each single context position within the contextual size of 10.	42
3.11	The F1-scores and number of model parameters obtained on the development and test sets using the feature selection at L543R345 when varying the PMI threshold from "baseline", -2, -1 and 0.	44
4.1	The proposed system architecture	46
4.2	A neural network consists of fully connected layers	50
4.3	A neuron in a fully connected layer	50
4.4	A single biLSTM layer.	51
4.5	The architecture of an LSTM cell [3].	52

List of Tables

3.1	Number of words and sentences in the dataset after pre-processing. Here, M stands for million, and k stands for thousand.	40
4.1	Dataset used to train word embeddings	54
4.2	Datasets used for SUD model training and evaluation	55
4.3	The confusion matrix between the system output and reference.	55
4.4	Experimental results for same-domain evaluation	57
4.5	Experimental results for the domain-mismatch evaluation	58
4.6	Evaluation results for the SUD models trained using Gigaword-Train, tested on both same-domain and domain-mismatch conditions	60

List of Abbreviations

ASR	Automatic Speech Recognition
CRF	Conditional Random Field
HELM	Hidden Event Language Model
HMM	Hidden Markov Model
LM	Language Model
PMI	Pointwise Mutual Information
MLP	Multi-layer Perceptron
NIST	National Institute of Standards and Technology
OOV	Out Of Vocabulary
SUD	Sentence Unit Detection
WER	Word Error Rate
DNN	Deep Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
SU	Sentence Unit
NSU	Non Sentence Unit
MaxEnt	Maximum Entropy
CBOW	Continuous Bag-of-Words

Chapter 1

Introduction

In a written text document, punctuations play an important role in representing and disambiguating the writer’s ideas in a sentence, a paragraph or the whole document. With the recent development in speech recognition research, a machine can generate a text document from the original speech audio, i.e. ASR systems, with high accuracy [4]. However, the generated transcript is not well structured, and is usually a stream of text without any punctuation, capitalization or paragraphing. This thesis aims to restructure ASR transcripts by developing a framework automatically detects and adds sentence units into the transcripts.

The next section presents the motivation for this thesis. Then Section 1.2 specifies the objective and scope. The contributions are summarized in Section 1.3, followed by the thesis organization in Section 1.4.

1.1 Motivation

1.1.1 ASR Transcripts

An ASR system receives speech audio as input and produces corresponding speech transcripts automatically. In recent years, ASR systems’ performance have improved significantly thanks to the availability of deep learning techniques. A report by Hinton et al. [4] shows that deep learning based ASR systems can reduce Word Error Rate (WER) by up to 8.9% compared to the conventional Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) when tested on spontaneous speech, e.g. conversational speech in Switchboard corpora, and user uploaded videos on Youtube. Many ASR systems

managed to achieve WER as low as or even better than human parity of around 5% of WER, for example Baidu’s Deep Speech 2 WER was 3.1% [5], Microsoft’s engine WER was 5.1% on conversational speech [6], Google’s speech model WER was 4.9%. Although these WERs obtained by the above organizations are not comparable because of different testing conditions, these results show that reliable ASR transcripts are now achievable.

Even though ASR systems have reached a certain level of reliability, ASR transcripts are typically unstructured. This results in difficulty reading ASR transcripts and prevents the usability of ASR transcripts in most downstream natural language processing applications, for example machine translation [7, 8], question answering [9, 10] and commonsense concept extraction [11, 12], which require well-structured documents for input. In order to improve readability and usability of ASR transcripts, or enriching ASR transcripts, involves many tasks. Examples of tasks to enrich ASR transcripts are illustrated in Figure 1.1. The original producing ASR transcript is a stream of words without any punctuation or capitalization. It also consists of meaningless words, such as noise marking and disfluencies. In the figure, the first enriching task is to remove the meaningless words, then sentence unit detection to break the stream of words into sentence units, and finally Capitalization to capitalize necessary words such as sentence beginning and name entities. It is clear that the final enriched transcript, is much more well-structured and easier to read compared to the initial transcript generated in the *Producing ASR transcript* step.

1.1.2 Sentence Unit Detection

Sentence unit detection is a part of the task of enriching ASR transcripts, it detects sentence units for the transcripts. The definition of sentence units in this thesis is slightly different from the common understanding of sentence units in written text. Here, it refers to either a well-structured grammatical sentence or a *semantically but not grammatically* complete sentence, which is similar to the sentence unit definition in Liu et al. [13]. SUD, which is sometimes referred to as sentence boundary detection, for ASR transcripts also differs from the task with the same name in the case of written text. SUD for a written document is limited to determining whether a given punctuation symbol, e.g. period and comma, indicates the end of a sentence or a clause. This is because the written document

Figure 1.1: Enriching ASR transcripts by (1) removing disfluencies and non-speech words, (2) sentence unit detection and (3) capitalization (including name entity detection).

already contains punctuation symbols but whether these symbols indicate the end of a sentence/clause might be ambiguous. In contrast, punctuation symbols are totally absent from ASR transcripts. Thus, this thesis considers the SUD task for ASR transcripts is to identify the location of sentence unit end-points in the transcripts.

In literature, SUD is usually a post-processing step, as depicted in Figure 1.2, which is performed after a transcript is generated by an ASR system. This is firstly because including SUD into an ASR system will significantly increase system complexity [14]. Secondly, a separate system might utilize audio-based information, e.g., prosodic features, more effectively.

Other enriching tasks that relate to SUD for ASR transcripts are sentence segmentation, punctuation restoration [15] and punctuation detection [16, 17]. They all aim to break an unpunctuated transcript into sentence-like units by putting sentence boundary

Figure 1.2: An overview of the SUD task as a post-processing step after the ASR system produces a transcript from the audio input.

labels, punctuation marks, etc. into the transcript.

SUD has had around two decades of development. The task was first clearly defined by the Stolcke and Shirberg team at the International Computer Science Institute (ICSI), University of California, Berkeley [18]. The team was then a part of research groups funded by the U.S National Research Foundation (NRF), under the DARPA Effective, Affordable Reusable Speech-to-text (EARS) program, with the aim to “significantly advance the state of the art in multi-lingual speech recognition of both broadcast news and human-to-human conversational speech”¹. SUD was one of the tasks involved under the EARS Rich Transcription project. Over the course of five years, from 2002 to 2007, the EARS program contributed significantly to the development of SUD and its related tasks. Later, the SUD task also received much attention through two famous evaluation campaigns, i.e. NIST Rich Transcription (from 2002-2007) and Internal Workshop on Spoken Language Processing (from 2009 till now). Over the course of SUD development a common SUD framework, typically used in previous studies, has been formed.

A common SUD framework is usually a combination of two major modules, i.e. feature extraction and modeling. The feature extraction module extracts relevant features that are then used in the modeling module as depicted in Figure 1.3. The framework workflow comprises two phases, i.e., training and testing. During the training phase, relevant features and parameters are learn to form a trained model. Consequently, the trained model is used for the prediction of sentence units on unlabeled transcripts during the testing phase. While the feature extraction module behaves the same in both phases, the modeling module behaves differently. It uses its model learning function during training and uses its prediction function during testing. Input transcripts to the framework are

¹<http://www1.icsi.berkeley.edu/Speech/EARS/>

also different in the two phases. The input transcripts require sentence unit labels in the training phase, while in the testing phase the labels are absent. The feature extraction module can be further broken down into two sub-modules, i.e., lexical feature extraction and prosodic feature extraction. This is based on the nature of the types of input data to a SUD system, i.e., whether they are transcripts (or text documents) that provide lexical information, or audio waveforms that provide prosodic information.

Figure 1.3: A common SUD framework.

Lexical information in SUD. Lexical information is a crucial knowledge resource for SUD [19], since it represents “what is being said” in a spoken document [19]. More importantly, it contains the grammatical structure of each sentence. Additionally, lexical information retains the correlations between individual elements within the sentence and between sentences within the document. Previous works showed that using only lexical features can result in a reasonable performance for SUD. In particular, Che et al. [17] achieved a F1-score of 78.8% when employing lexical features in a Convolutional Neural Network (CNN)-based model. By using a combination of multiple lexical features for a bi-directional Long Short-Term Memory (LSTM), Xu et al. [20] achieved a F1-score of 82.5% on Broadcast News reference transcripts. A neural machine translation SUD system using lexical features by Klejch et al. [21] achieved F1-scores of up to 87.03% when tested on reference transcripts of a multi-genre Broadcast News corpus. There are two major advantages in using lexical information for the SUD task. Firstly, it is

derivable from the readily available written text sources, e.g., web pages, online lecture notes and e-books. Secondly, such lexical-only SUD systems require significantly lesser storage and computational resources as compared to those with combined prosodic and lexical information.

Although previous studies have utilized a number of methods for feature extraction and modeling, there is still a lack of attention to feature selection methods that are specialized for the SUD task. This report therefore attempts to tackle this problem with the aim of improving existing SUD systems by fine tuning their input features, that to reduce input features and model complexity and to improve the system performance by suggesting the most dominant variant among multiple variants of a same feature types.

1.2 Objective and Thesis Scope

This thesis considers the SUD task as a sequential tagging task, in which given a word sequence, the task is to label each word boundary in the input sequence with either a Sentence Unit (SU) label or a Non Sentence Unit (NSU) label to represent the presence or absence of sentence boundary after the word. Figure 1.4 demonstrates the output of the SUD task when it works as a sequential tagging task.

Figure 1.4: An example of a SUD as a sequential tagging task.

Furthermore, this thesis uses lexical information as the sole knowledge resource to the SUD systems. This is because of the many advantages of using lexical information. The first advantage is that the lexical features are easy to extract from numerous available sources. Secondly, the building of lexical-based SUD models requires lesser computing and storage resources as compared to the the building of multiple-source SUD models.

1.3 Contributions

This thesis studies two approaches to improve the lexical input feature space for SUD systems. The first approach proposes a PMI-based feature selection method to select only informative features for a shallow learning (CRF-based) SUD system [22]. The second approach examines various strategies to train word embedding models which are used to extract feature for a deep learning based SUD system [23]. The contributions of the thesis can be summarized as below:

The novel PMI-based feature selection method [22]: The approach investigates the effect of contextual features to the SUD task and proposes a feature selection method using PMI to reduce the input feature space. Here, the surrounding words of each label being predicted are used as the contextual features to the SUD system. Experimental results show that increasing the size of surrounding context gradually from two to ten words improved the system performance. The performance is optimum when the size reaches ten words, i.e., five preceding words and five following words for the label being examined. In addition, the study found that the nearer context, i.e., the first and the second preceding words and the first and the second following words, are dominant for the task. The ones farther away contain redundancy. Based on these findings, the proposed PMI-based feature selection method are demonstrated its capability to significantly reduce redundant features for far context, while still maintaining a system performance comparable to the baseline model.

Examining word embedding representations for a deep learning based SUD [23]:

The approach examines four variants of word embedding representation. They are

trained with word-based [24] and subword-based skip-gram [25] using in-domain and out-of-domain datasets. The SUD models trained with word embedding vectors extracted from the four variants were evaluated on three testing conditions: (1) test with reference transcripts from the same-domain dataset, (2) test with reference transcripts from the domain-mismatch dataset and (3) test with ASR transcripts from the domain-mismatch dataset. The findings are: (1) the in-domain subword-based embedding gives the best results for all testing conditions. (2) When the out-of-domain word embeddings were used, word-based embedding was shown to be better than subword-based embedding. (3) Even though a large training dataset was used, i.e., the broadcast news speech transcripts contain more than six million of words, system performance was still significantly decreased when tested on domain-mismatch conditions. Performance also decreased when tested with ASR transcripts contains 29.5% WER.

The first work was published in the proceeding of the 8th Asian Conference on Intelligent Information and Database Systems (ACIIDS) in 2016 [22]. The second work has been accepted for presentation in the 2018 International Conference on Asian Language Processing (IALP 2018) [23].

1.4 Thesis organization

The rest of this thesis is organized as follows:

Chapter 2 is a literature survey of the SUD task. The chapter presents an overview of the task from different aspects, i.e. feature extraction and modeling approach.

Chapter 3 presents the proposed PMI-based feature selection method that reduces the space of input features without adversely affecting the performance of a CRF-based SUD system.

Chapter 4 presents the proposed approach to examine the effect of using different word embedding types to a deep learning based SUD system.

Chapter 5 provides a conclusion to the thesis, summarizes our contributions and discusses potential directions for future work.

Chapter 2

An Overview of Sentence Unit Detection for ASR Transcripts

This chapter provides an overview of sentence unit detection specifically for ASR transcripts. In particular, section 2.1 briefly reviews the common work-flow of a SUD system consisting of feature extraction and modeling modules. Subsequently, section 2.2 and section 2.3 review feature extraction methods and modeling techniques respectively, that are popularly employed in literature. Finally, section 2.4 concludes the chapter and introduces the motivation for the next chapters.

2.1 Sentence Unit Detection

In this work, Sentence unit detection is defined as the sequential tagging task that aims to predict a sequence of SU or NSU labels corresponding to each word boundary in an unpunctuated input text sequence. A SUD system's work-flow consists of feature extraction and modeling modules. Given a raw audio waveform and its unpunctuated transcript, feature extraction is first performed to extract either lexical or prosodic or both information from speech. Lexical features refer to features extracted from the transcription text, e.g., n-gram [18], Part-Of-Speech (POS) tags [19, 26] or word embedding vectors(word2vec) [20, 27]. Prosodic features are derived from the audio data such as pauses, energy and pitch. Extracted features can then be used by any appropriate modeling techniques to train SUD models and to detect SU labels. Typical examples of machine learning techniques for SUD include HMM, CRF, Deep Neural Network (DNN) and LSTM.

A detailed review of typical features used is presented in the next section, followed by section 2.3 in which the common machine learning techniques are discussed.

2.2 Features Extraction

Many features can be extracted from the original audio and transcripts that are given as input for a SUD system. Typically, these features can be classified into two categories, i.e. lexical and prosodic features. Lexical features represent “*what is said*” while prosodic features represent “*how it is said*” [19]. The following subsections describe these features and methods to extract them in more details.

2.2.1 Lexical Features

Lexical features are the most popular features for the SUD task [19, 28, 29]. This is due to many reasons, e.g. the task was inspired by the success of the same named task applied on written text [19]. Additionally, in some scenarios, lexical information is the only available information. For example, in the evaluation campaign of the International Workshop on Spoken Language Translation in 2009 (IWSLT 2009) [30], the data provided consisted only of transcripts without original waveforms [28]. Also, textual data is massively available from the Internet and there exist many highly accurate natural language processing toolkits, which can be used for lexical feature extraction. Thus, extracting and using lexical features for the SUD task has a very low barrier to entry.

Language Modeling. Previous studies have mentioned n-gram, skip-gram, bag-of-words, etc. as lexical features in their systems [15, 18, 19, 26]. These features are inspired by techniques used in language modeling. A Language Model (LM) captures the syntactic, semantic, and pragmatic properties of text documents based on statistical principles. It can generate a probability measure for a given sequence of words. For example, given a sequence of m words, $W=w_1, \dots, w_i, \dots, w_m$ ($1 \leq i \leq m$), an LM can produce a probability of all the m words sequentially appear together, i.e. $P(w_1, w_2, \dots, w_m)$.

There are several types of LM features, e.g. n-gram, distant bi-gram, skip-gram, trigger and bag-of-words. The two major differences between them are the maximum length of historical context they can cover, and the ability to preserve the word ordering

Figure 2.1: An example of common feature sets used for LM[1].

within the context [1]. Figure 2.1 shows a few examples of language modeling variations, i.e., n-gram, distant bi-gram, skip-gram, trigger and bag-of-words. Although n-gram is able to preserve of all order in the context, it is usually limited to only cover the shortest context ($n \leq 4$) compared to the others. Distant-bigram and skip-gram select only a few words within the context, and thus they are able to cover a longer context as compare to the n-gram. In contrast, trigger and bag-of-words are totally relaxed on the word ordering, and thus they can handle the longest context ($n \geq 10$). In practice, these variations of language modeling features are used together as they are complimentary.

Figure 2.2: An example of n-gram used to predict SU labels.

Language Models used in SUD. LM techniques can be applied as a prediction model for the SUD task. For example, a typical n-gram model can be utilized to predict the SU labels by casting them as tokens in its lexicon [18]. Figure 2.2 explains a method of predicting SU labels in N-gram LM realization with $N = 4$. Using a trained 4-gram

model, we can evaluate the probability of observing the target label “SU” versus “NSU” given the preceding three words context of “ok”, “NSU” and “you”. Stolcke et. al. [31] proposed to treat SU and NSU labels as hidden events in a LM model, namely as the Hidden Event Language Model(HELM), to jointly capture the probability of a word in the given sequence and its SU and NSU labels. The success of using n-gram and HELM in predicting SUs has been utilized in SRILM¹ - a popular LM toolkit developed by Andreas [32]. Its SUD function has also being applied for machine translation and is a part of the MOSES toolkit² - a machine translation toolkit - to segment a document into sentences and translate it from one language to another [7].

The other way to utilize LM techniques in SUD is to treat them as methods for feature extraction. The extracted features are the input sequence with probability distributions over the given training data. This extraction could be processed via a number of available NLP toolkits such as SRILM and NLTK³. Feature extraction using language modeling techniques can also directly perform in Markov style models such as HMM, Maximum Entropy (MaxEnt) and CRF [33].

Word embedding. Word embedding is a relatively new form of word presentation produced by neural network based LM [34]. Instead of a statistical probability distribution over n-gram of words, a neural network based approach models the distributed representations of words. Specifically, a word is represented by a distribution of weights across a dimensional vector which reflects the relationship between the word and its context neighbors.

Two popular word embeddings, i.e. Continuous-Bag-Of-Words (CBOW) and skip-gram, were proposed by Mikolov et al. [35]. They are extracted from two neural language model architectures which reflect the relation between a word and its neighbors. The CBOW network predicts the current word based on its neighbors, while the skip-gram network predicts the neighbors given the current word[35]. Figure 2.3 depicts these two architectures with an example of the current word w_t =“out” and its neighbors, i.e. w_{t-2} =“we”, w_{t-1} =“go”, w_{t+1} =“for”, w_{t+2} =“lunch”. The CBOW predicts the current word w_t using the neighbors, while the skip-gram predicts the neighbors

¹<http://www.speech.sri.com/projects/srilm/>

²<http://www.statmt.org/moses/>

³<https://www.nltk.org/index.html>

Figure 2.3: Examples of CBOW and skip-gram representations.

$w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ based on w_t . The CBOW and skip-gram word embedding can be efficiently trained and extracted on a customized training text data using the word2vec toolkit⁴ provided by the same authors.

Word embedding variations are popularly employed in recent neural network based SUD models. For example, Che et al. [17] used embedding vectors built using GloVe [36], Xu et al. [20] used supervised word embeddings, where the word embeddings are trained toward the targets of SU or NSU labels. Klejch et al. [21] trained their word embedding using Theano [37].

NLP tags. NLP tag based features such as POS, Chunk tags are popular for the SUD task [26, 38, 39]. The tags are known to carry the grammatical, linguistic characteristics of words in a sequence. POS tags are typically assigned to each individual word in a sequence, to indicate its grammatical form, such as whether the word is a noun, a verb or a preposition. On the other hand, Chunk tags work on a group of a few neighboring words, where the group is a “chunk” of grammatical phrase, e.g. noun

⁴<https://code.google.com/archive/p/word2vec/>

phrase and verb phrase. For example a “chunk” of noun phrase that contains three individual words, the words will be sequentially tagged as Start-of-Noun-Phrase (S-NP), In-Noun-Phrase (I-NP) and End-of-Noun-Phrase (E-NP).

Input:	it	is	a	beautiful	day	should	we	go	out	for	lunch
POS:	PRP	VBZ	DT	JJ	NN	MD	PRP	VB	RP	IN	NN
Chunk:	S-NP	S-VP	B-NP	I-NP	E-NP	O	S-NP	S-VP	S-PRT	S-PP	S-NP

POS tags:	<ul style="list-style-type: none"> PRP: Personal Pronoun VBZ: Verb, 3rd person singular present DT: Determiner NN: Noun, singular or mass IN: Preposition or subordinating conjunction 	<ul style="list-style-type: none"> JJ: Adjective MD: Modification VB: Verb, base form RP: Particle 	Chunk tags:	<ul style="list-style-type: none"> S-NP: Singular Noun Phrase S-VP: Singular Verb Phrase B-NP: Begin Noun Phrase I-NP: In Noun Phrase S-PRT: Singular Particle 	<ul style="list-style-type: none"> E-NP: End Noun Phrase O: Other S-PRT: Begin Noun Phrase S-PP: Singular Preposition
------------------	---	--	--------------------	---	---

Figure 2.4: An example of an input sequence with its POS and Chunk tags.

In the past, NLP tags such as POS and Chunk tags were only obtainable through manual tagging by linguistic experts. Recently, the tags have been accurately obtainable through machine learning techniques. Famous toolkits that perform well on these tagging task are SENNA toolkit [40, 41], NLTK⁵, the Stanford Log-linear POS Tagger [42, 43] and SpaCy⁶. An example of a POS and Chunk tags produced using SENNA is shown in Figure 2.4. The availability of these toolkits contributes significantly to the usability of NLP tags in SUD, since using available pre-train models no longer required manual work.

Previous studies, e.g., [17, 26, 31, 44, 45, 46, 47], have shown that language model variations, e.g., n-gram, word embedding, and NLP tags are informative to the task. This is based on the nature of written (and also for spoken) documents, in which words in the same sentence always have a certain level of cohesion, as opposed to words from different sentences. Stolcke et al. [31] has shown that improved performance was achieved when POS tags were added as features to their existing HMM which relied on an underlying n-gram language model. Consequently, Huang and Zweig [44], Liu et al. [45] and Xu et al. [26] also employed POS and Chunk tags in their systems. Experimental results

⁵<https://www.nltk.org>

⁶<https://spacy.io>

in the above works showed improvement when the tags were added as additional input features. More recently, word embedding with variations were commonly used in deep neural network based systems [17, 21, 47]. Word embedding has shown to be more effective as compared to traditional n-gram styled features because of two main reasons. Firstly, it does not face the problem of data sparsity. Secondly, it does not require a fixed context length and thus can be dynamically set to better fit the input data.

Lexical features have been shown to be informative for the SUD task. They are easy to obtain through a number of available toolkits. Additionally, most of the available toolkits, e.g., SENNA [40], word2vec [35] or GloVe [36], allow their models to be re-trained with customized training text. On the other hand, text data sources are readily available on the Internet, through various sources such as online news, lectures, e-books, Wikipedia⁷. These advantages inspired this study to only focus on using lexical features and to extensively explore the effectiveness of different lexical features for the SUD task.

2.2.2 Prosodic Features

In this thesis, prosodic features refer to prosody properties of larger units, e.g., a syllable, a word, or a utterance of speech rather than individual phonetic segments, such as vowels and consonants, which might be more common in other research. Prosody is known to be able to preserve descriptive characteristics of speakers' emotions, focus and their intended sentence end forms, e.g., question, statement or command. Because of the last characteristic, prosodic features are considered as exclusive and descriptive information sources for SUD for ASR transcripts [2, 14, 31, 48].

Hundreds of prosodic features have been used in past studies [26, 31, 49]. However, there does not exist a countable list of prosodic features in literature [50]. It is only agreeable that prosodic features can be described via their two common aspects, i.e., auditory information and acoustic information. Auditory information is more related to individual speaking styles, .e.g., the pitch of a voice, lengths, loudness and sound quality. Acoustic information on the other hand relates to computationally extractable information such as fundamental frequency (F0), duration, intensity and spectral representation. In SUD, prosodic features employed are commonly derived from acoustic information rather than

⁷<https://www.wikipedia.org>

auditory information. The section below briefly describes three major prosodic feature groups within the category of acoustic information, i.e., pause duration, energy and fundamental frequency (F0). Most of features used in recent SUD systems [26, 39, 49] are derived from these three.

Pause Duration: Speakers tend to have longer pauses than usual at the end of a sentence [31, 49]. This holds true not only in English but in other languages such as French, Japanese, Hindi, Spanish [51]. They are described as voiceless regions within a stream of continuous speech which fall below an adaptively found threshold [2]. Figure 2.5 shows an aligned speech signal and word level transcript of the given speech utterance. In this example, there is a long pause at both of the word boundaries which are tagged with SU labels, especially after the word “day”. However, pause duration and distribution might vary in different languages and are speaker dependent [49]. This

Figure 2.5: An aligned speech signal and word level transcript for a spoken utterance. Clear long pauses are observed at the sentence ending word boundary, i.e., “day” and lunch.

variation might even occur from the same speakers but in different speaking conditions, e.g., between reading and spontaneous speech. This means that pause durations might require a normalization before being utilized as features for SUD systems.

Fundamental frequency: Fundamental frequency (F0), sometimes referred to as pitch - although they are not exactly the same - is another popular prosodic feature used in SUD. F0 represents the frequency of speech. It is drawn from the lowest frequency

values of a periodic waveform. It has been observed that variations of F0 feature, i.e. F0 range, F0 reset, F0 slope and F0 duration are particularly informative for SUD [49, 51]. Figure 2.6 shows an aligned diagram of F0 variations with word level transcripts of a given speech utterance. It shows the fluctuation of F0 values over the speech signals along the time axis. Here, the same observations as in previous works [2, 49, 51] were noted: Firstly, there is a tendency for F0 to slope downwards at words with SU labels, i.e., “day” and “lunch”. Secondly, broken trajectories happen at both sentence ends, i.e., “day” and “lunch”. And lastly, there is sign of pitch reset at the beginning word of the sentence, i.e., “should”. It is also important to note that raw F0 features are difficult to model [49], and thus might require smoothing techniques when used.

Figure 2.6: Examples of F0 features: F0 slope, F0 Continuity, F0 reset and F0 range.

Energy: Energy reflects the power used in a spoken utterance, i.e. a loud voice is represented by higher energy values while a soft voice is represented by lower energy values. The usefulness of energy features for SUD is still controversial. Shriberg et al. [49] claimed that energy features are less reliable and largely redundant with pause duration and pitch features. However, Pausch [2] observed that speakers generally finish long sentences with lower energy rather than taking an additional breath towards the end of the sentence. Thus, the author concluded that measuring the decay of short-term energy can be quite informative for the SUD task. Figure 2.7 depicts energy decay features. The figure is from Pausch’s work [2].

From these three main features, it is possible to extract hundred of variations. Examples of *pause duration* variations are: Pause durations after words, phonemes, segments and speaker turn [52]. *F0 variations* include F0 slope, F0 contour, F0 continuity, F0

Figure 2.7: Frame-based energy progression of a speech signal with energy decay (red) in [dB/s] within potential sentence boundaries (green) [2].

reset [49], F0 at downdrift, initial and final intonation [2]. And *energy variations* are: energy decay, energy pause duration and energy intensity [2]. Others than these three main groups, previous studies have also mentioned auditory features [39], speaker gender, speaker turn [49], rate of speech (ROS) and frame-based features such as frame energy, zero-crossing rate [53].

One way to extract prosodic feature is to use open-source prosodic extraction toolkit. A prosodic features extraction toolkit is provided by Huang et al. [52]. The toolkit can extract multiple prosodic features without requiring a lot of hand-labeled data [52]. This makes prosodic features easily extractable from audio sources. More importantly, prosodic features are relatively robust with respect to recognition errors by the ASR system, as extracting these features does not rely on the detected words. Thus, prosodic features are considered as “*ASR error tolerant*” features in literature [31].

It was not popular to use prosodic features as the sole feature source to a SUD system in previous studies. However, studies by Wang et al. [53] and Wang and Narayana [48] showed that using only prosodic features still managed to achieve remarkable results. In particular, the system by Wang et al. [53] used pauses, ROS and a few phone-based features achieved 82.3% accuracy on conversational speech corpus. The system by Wang and Narayana [48] achieved 75% accuracy on a conversational telephony speech corpus

with only pitch features. A more common strategy for utilizing resources to a SUD system is to combine both lexical and prosodic features [16, 33, 45, 49, 54].

Even though we have provided an overview of prosodic features, this thesis choose to focus on only lexical information for SUD as explained in Chapter 1.

2.3 Modeling Approaches

Features obtained through feature extraction module are used for modeling. In order to model extracted features, there are two aspects to consider. The first is how to incorporate the features efficiently and the second is which machine learning technique to choose. This section reviews strategies and machine learning techniques that are common in literature.

2.3.1 Feature Incorporation Strategy

How to incorporate multiple available features efficiently is the first consideration in modeling module. It is straightforward to individually model single source features in a single model, i.e., prosodic model and lexical model, and then use the model for prediction. However a majority of previous works employ the combination of the two models for their systems [13, 20, 29, 55] as the two models are known to be complimentary to each other. Figure 2.8 summarizes a number of feature incorporation strategies commonly used in previous studies. In this section, two popular feature incorporation strategies are described. The first directly incorporates all available features into a single model (called Direct Combination). The second, considerably easier to deal with than the first, forms separate models for each feature sources, then combines the obtained models using some interpolation strategies (called Model Interpolation).

Direct Combination. Direct combination is a straightforward strategy for the incorporation of features into a modelling method. However, in practice, it faces a number of difficulties when features are from different sources with different characteristics and complicated dependencies. For example, features that correlate to the others will ruin the independent assumption in many modelling methods, while different features might require different representation forms that cause difficulty in handling all of them at once.

Figure 2.8: Group of modelling approaches based on feature type used

A popular approach of using a statistical finite state machine proposed by Christensen et al. [56]. In the work, The work is similar to the SUD task defined in this thesis that the goal is to predict the sequence of punctuation marks corresponding to the word boundaries of the input words sequence. The prosodic features used are only pause durations, phoneme durations and pitch. For lexical features, they employ an LM-based model providing the joint probability of a sequence of words with their corresponding punctuation classes. Both prosodic and linguistic features are experimentally modeled using two approaches, i.e. a statistical finite state machine (FSM) and a Multiple Layer Perceptron (MLP). Experimental results confirm that models from combined linguistic and prosodic features achieve better performance as compared to the models that use only lexical information. Moreover, the author concluded that pause durations are the most informative prosodic features for the task.

Another attempt at the direct modeling approach is the work by Huang and Zweig [44]. The authors propose a MaxEnt framework that directly models lexical features, i.e. n-gram with context of five words which contains two preceding and two following of the word being examined, and only pause duration as a prosodic feature. Experiments on spontaneous speech corpus show that their proposed framework achieve a F-measure of 80% and 73% on reference and automatic transcript, respectively. More recently, Tilk and Alumäe [47] introduce an LSTM model for punctuation restoration task. In the

work, the lexical features are word embedding vectors obtained using an LSTM model with attention mechanism for the punctuation restoration task. The lexical features are then concatenated to pause duration features as input to another LSTM model for final prediction. The model shows an overall reduction of 1.37% and 1.01% classification error rate on reference and ASR transcripts respectively, when compared with a “*lexical-only*” LSTM model.

Model Interpolation. The second approach is to interpolate the results from two individual models of lexical and prosodic features. Common flow for these combined systems is: Firstly, separate single models for which lexical and prosodic features are built; Then, a combination method is used to interpolate the two models, forming the final model for prediction. Alternatively, a lexical model is first built, and its prediction probabilities are then incorporated into the prosodic models, and vice versa, the prosodic model provides posterior probabilities to incorporate into the lexical models. The latter is more common than the former in literature.

A study by Gotoh and Renals [29] shows that the combination of a LM model and a pause duration model significantly outperforms the baseline n-gram models. Shriberg et al. [49] builds a separate n-gram language model for lexical features and a Decision Tree for prosodic features. The model is then combined through a HMM framework. Another example is the CRF-based system proposed by Liu et al. [45]. In the work, lexical features are directly modeled by CRF, but prosodic features are modeled in a DT. Preliminary prediction results from the DT were then put into CRF as additional features. Inspired by Liu et al. [45] work, Xu et al. [26] built a similar system, however, instead of DT, the authors used a deep learning method to handle prosodic features.

In summary, a directly combined model is the preferred choice of modeling when dealing with limited features, while model interpolation is better suited when large number of features are involved. In the next section, a review of machine learning techniques that are popularly used in the literature is presented.

2.3.2 Machine Learning Techniques

Over the last two decades, a substantial amount of machine learning techniques have been applied for the SUD task. Figure 2.9 summarizes the taxonomy of machine learning

techniques for SUD by organizing the techniques into different categories, i.e. supervised learning, semi-supervised learning and unsupervised learning. The largest category is the supervised learning, which includes both shallow and deep learning (will be explained shortly) techniques, and the next category is semi-supervised learning. There are an unknown number of unsupervised learning applied to SUD so far.

Figure 2.9: A top-down tree map for modelling approaches in SUD research.

Note that in this thesis, the techniques under shallow learning and deep learning are distinguished by:

- (i) Shallow learning techniques rely heavily on feature engineering and selection. Deep learning techniques are almost able to learn feature representation themselves.
- (ii) Shallow learning techniques are graphically represented by networks that contain no or only one hidden layer while deep learning have neural network structures, with more than one hidden layer.

In the following sub-sections, the use of common techniques from the above three major groups, i.e. shallow and deep, and semi-supervised learning, in various of previous works will be summarized.

2.3.2.1 Shallow Learning

Shallow learning refers to techniques that have graphical representation with less than 1 hidden layer, consisting mainly of directed and undirected models, i.e., HMM, MaxEnt and CRF. The use of these three techniques in SUD will be reviewed in the following sections.

Hidden Markov Models. HMM is a statistical model that relies on the Markov chains theory. It assumes that the input system is a Markov process containing hidden states. Each state in a HMM may connect to other states by an edge. The transition between states is regularized by state transition probabilities. In SUD, a simplified version of HMM, i.e., first order HMM, is usually employed.

A very first HMM-based SUD framework is proposed by Shriberg et al. [49]. The framework provides the capability of handling both lexical and prosodic features. Specifically, HMM serves as an interpolation method, to incorporate prior results from lexical and prosodic models. The lexical model is built using a statistical LM method and the prosodic model uses a DT. The resulting model outperforms the state-of-the-art statistical n-gram models. The proposed HMM based framework is then popularly adopted in multiple participating systems of NIST Rich Transcription Evaluation in Fall 2003 [38]. The only difference among the proposed systems is feature choice and the method to incorporate the features into the framework. Features are modeled by HMM either through direct combination, or model interpolation. Through the evaluation, the effectiveness of HMM with respect to the SUD task is confirmed, i.e., all the winning systems in different evaluation conditions are built using a HMM based framework.

It is generally agreed among researchers that HMM is effective for the SUD task [33, 38, 49]. This is because HMM has a number of advantages over the statistical LM methods. Firstly, it is able to capture the context of the sequence. Secondly, it is computationally efficient, and thus it is easy to incorporate real-value features, e.g., posterior probabilities produced by prosodic models [49]. However, a drawback of HMM is that it maximizes the joint probability of training observations and events, resulting in decreased performance from unseen observations when performed under test conditions. Additionally, its underlying LM strongly depends on the assumption of feature independence, therefore it faces difficulty in incorporating multiple correlated features [33].

Maximum Entropy Models. To overcome the drawbacks of HMM, MaxEnt is proposed in 2002 [44]. MaxEnt relies on conditional likelihood theory, i.e., it directly models the labels posteriors and explicitly discriminates between correct and incorrect labels.

Two remarkable SUD systems relying on the MaxEnt framework were quite successful in the past: the Huang and Zweig [44] system and the the Liu et al. [33] system.

These systems made use of the flexibility of the MaxEnt model when incorporating multiple features, especially the correlated ones. The MaxEnt system [44] were designed for punctuation annotation on conversational speech transcription. The input features are five neighboring words and pause duration measured in 0.01 second intervals. These features are inputs to the uni-gram and bi-gram models. In the Liu et al. system, the MaxEnt was investigated with a richer set of lexical and prosodic features, including n-gram, POS, speaker turn, pause duration, pitch and energy. Because the independent assumption was relaxed, multiple lexical features were engaged directly in the MaxEnt model together with posterior probabilities from a DT-based prosodic model. Experimental results showed that the MaxEnt model outperformed the HMM model when dealing with only text features. However, when both prosodic and lexical features are used, the MaxEnt model performed worse on automatic transcripts. Interestingly, the combination of the two models yields the best performance. This is reasonable as the two approaches are complementary in strengths and weaknesses. The MaxEnt model can combine several knowledge sources with ease and the HMM model is capable of directly modeling real-valued prosodic features.

The MaxEnt model therefore provides the advantage of being able to model multiple features easily. However, it has two drawbacks: 1) it models only local evidences, i.e., surrounding words and prosodic features, and 2) it can only receive categorical features. The first drawback means that longer context information is not captured, while the second drawback means that continuous real-valued data, which is frequent in the SUD task, needs to be transformed to suit the input requirements of MaxEnt, leading to a loss of information.

Conditional Random Fields. CRF is proposed by Lafferty et al. [57] and further optimized for shallow parsing and similar tasks by Sha and Pereira [58]. CRF has a combined advantages from the HMM and MaxEnt models, and thus has been increasingly employed in the SUD task. Similar to MaxEnt, the CRF model can integrate multiple correlated features easily. Additionally, the CRF uses sequence decoding that is globally optimal, which is one of advantage of HMM.

A common variation of general CRF, i.e., Linear Chain CRF, was first adopted for SUD by Liu et al. [45]. In which, the proposed framework includes two steps: A DT

is employed to model prosodic features. Its predicted posterior probabilities were then categorized as input for the CRF model. On the other hand, lexical features were word identities, POS tags and Chunk tags. The work shows that the CRF framework is superior to both HMM and MaxEnt, especially for conversational speech corpora. The work also confirms that the combination results from HMM, MaxEnt, and CRFs yielded the best result in all testing conditions.

Among various of shallow learning techniques, CRF was found to be superior the others[26, 39, 45]. Thus, we adopts a CRF framework that similar to Liu et al. [45] and Xu et al. [26] systems as the baseline system to our work presented in Chapter 3.

2.3.2.2 Deep Learning

The very first deep learning architecture adopted for SUD by Xu et al. [26]. In the work, a Deep Feed-Forward Neural Network (DNN) is used to model prosodic features and produces posterior probabilities as input for a CRF framework for further modelling. Since then, various deep learning architectures have been used such as CNN [17], Recurrent Neural Network (RNN) [47] and LSTM [21]. This section briefly reviews the above mentioned deep learning architectures for SUD.

Deep Feed-forward Neural Network. In 2014, Xu et al. [26] made the first attempt at using deep learning, i.e. DNN, for SUD. Specifically, the work further improved the Liu [19] CRF based framework by replacing the DT with a DNN architecture to model prosodic features. The DNN transformed prosodic feature inputs into a non-linear mapping of posterior probabilities of boundaries. The probabilities were then combined with lexical features as input to a linear-chain CRF model for final prediction of sentence boundaries. As compared to DT, DNN is able to capture more complex statistical patterns. This is because DNN can learn important feature representations from a large raw input set through non-linear transformations. Thus, a DNN based model is more suitable for dealing with real-valued prosodic features. Even though both DT and DNN experience deterioration of performance on ASR transcripts as compared to reference transcripts, the DT model suffers more performance deterioration, indicating that DNN is more robust in handling “imperfect prosodic features”. Experiments on Broadcast News corpus

showed that the DNN-CRF model significantly outperforms the DT-CRF model. Specifically, the DNN-CRF model reduced NIST scores by up to 16.7% and 4.1% on reference and automatic transcriptions respectively as compared to the DT-CRF model.

Convolutional Neural Network. CNN is a specialized deep learning architecture for dealing with image processing problems. Very deep CNN architectures were very successful in a number of recent image processing competitions. For example Resnet He et al. [59] won first place in several tasks in the ILSVRC2015⁸ competition and the COCO 2015 competition⁹.

CNN is rarely adopted for the SUD task, as its fundamental concept make it different from the task, i.e. CNN is specially designed for grid-based data, while input to an SUD system is sequential. However, there is an interesting work using CNN for SUD by Gale and Parthasarathy [60], in which the authors employed a very deep CNN network to produce character-based features. These features are then fed into a bidirectional LSTM (biLSTM) architecture, in conjunction with word embedding features for predicting punctuation. Experiments showed that the character-based features built from CNN were useful to the biLSTM model.

Long Short-Term Memory. Recently LSTM has been shown to be effective for sequence tagging tasks. This is because LSTM is able to capture and remember historical context information, which is crucial for sequential tasks.

In 2015, Tilk and Alumae [47] introduced a two-stage LSTM for the SUD task. This was the first study applying RNNs for punctuation restoration. LSTMs exhibit many characteristics that make them more suitable for the SUD task compared to primitive models like n-grams. LSTMs overcome the data sparsity issue and generalize well to unseen data instances by learning distributed word representations [34] or entire contexts [35]. The biLSTM based SUD achieved a 16.9% and 8.8% reduction in the number of restoration errors for reference and ASR transcriptions respectively when the LSTM model was tested on Estonian broadcast news, compared to a DT model that uses HELM posteriors and pause durations.

Recently, Xu et al. [20] proposed a biLSTM based framework for SUD. The significant difference of the work done by Xu et al. [20] as compared to other biLSTM based

⁸<http://image-net.org/challenges/LSVRC/2015/>

⁹<http://cocodataset.org/#detection-2015>

systems is that the work used a supervised word embedding instead of unsupervised word embeddings. The work shows that the word embedding vectors, which are obtained from supervised training on a large dataset with attention to sentence unit labels, were more closely correlated to the presence of the SU labels. The authors compare two SUD models, the first model uses the unsupervised CBOW and the second model uses the supervised embedding vectors as the lexical feature. The experiment results on a Broadcast News corpus shows that the latter significantly outperforms the former. These results give an insight on a possibility to improve deep learning based SUD systems' performance by a careful choice of word embedding types as input for the systems. Thus, it is the motivation for our work, which will be later presented in Chapter 4, to perform an intensive examination of word embedding types usage for a deep learning based SUD system.

2.3.3 Semi-supervised Approaches

Other than the above supervised learning method, there were some previous works that attempted to deal with limited training data condition. Proposed approaches include model adaptation [61] and co-training [62].

The fundamental idea of model adaptations is to employ out-of-domain labeled data to build a model for the limited in-domain data. Cuendet et al. [61] proposed an adaptation approach in which a large amount of available conversational telephonic speech data was employed as out-of-domain data to perform SUD on a limited meeting speech corpus. The authors tried four adaptation strategies. In the simplest strategy, that of data concatenation, the classifier is built on the joined set of both in-domain and out-of-domain data. The second strategy is logistical interpolation, in which the output of two separately built classifiers is combined using a logistic function. The third strategy is to build a classifier on out-of-domain data and employ the confidence scores from this classifier as one feature in the in-domain classifier. The last strategy is a boosting adaptation strategy in which the classifier is built on out-of-domain data and the in-domain data is used to “adapt” the model using a boosting method. In general, logistical interpolation performed better than other adaptation strategies, independent of the size of the in-domain training data. It was observed that with an increase in the amount of in-domain meeting data used, there was a decrease in performance difference between

the model trained only with meeting data and the mixed models. Interestingly, apart from the data concatenation strategy, all adaptation strategies improved performance from the baseline model built only on meeting data, even with the use of the full meeting data. As preliminary work on applying model adaptation for SUD, the paper showed potential for more accurate adaptation strategies to increase performance or extend the proposed approach to other corpora such as broadcast news.

Another strategy to deal with limited training data is co-training. Co-training is a semi-supervised algorithm that uses multiple weak models to form a stronger model. Its objective is to increase the amount of labeled training data by incorporating large amounts of unlabeled data. Classifiers trained on different “views” of labeled data are used to label the unlabeled data. Out of this set of newly labeled data, the most confidently labeled examples are added to the manually labeled set. Several iterations of co-training are usually performed. Guz et al. [62] proposed to use a co-training algorithm to strengthen the SUD task by considering lexical and prosodic features as two views of the data. According to them, the amount of labeled data readily available for SUD justifies the requirement of the co-training approach. Their experimental results show that the co-training approach is superior to using only one single knowledge source (lexical/prosodic). Additionally, when the available training dataset is small, in the range of 1000 labeled samples, the co-training approach significantly outperforms the traditional supervised training baseline for both lexical and prosodic features by 25% and 12% relative.

2.4 Conclusion

This chapter has provided an overview of the SUD task in the literature. A broad description of the overall structure of an SUD system has been presented.

Many of the previous studies have been reviewed with regard to their feature extraction methods and modeling approaches used. Throughout this review, it can be concluded that lexical information is crucial for SUD research. Furthermore, there is a much less studies on optimizing of lexical features, such as feature extraction and feature selection, for the SUD task compared to proposing new modeling approaches. This

motivates us to focus on optimizing feature extraction and selection methods to improve the current SUD systems. Details of the work will be presented in Chapter 3.

The literature survey also showed a recent trend in SUD research which replacing shallow learning techniques by deep learning techniques in SUD. Different from shallow learning techniques that rely heavily on feature extraction process, many deep learning techniques are known to be able to extract feature representation from raw feature efficiently. This open a question of whether optimizing input feature space still helpful for deep learning based SUD models. This question motivates our work which will be presented in Chapter 4.

Chapter 3

Optimizing Input Feature Space Using Pointwise Mutual Information

In the last chapter, an overview of previous works and a review of state-of-the-art feature extraction and modeling techniques were presented. The review revealed that many existing systems did not perform feature selection prior to using them for modeling. Without selection could significantly increase computational cost and storage space requirements of models. These two issues motivate this work to examine PMI as a feature selection technique. PMI measures the relevance between an input feature and the presence of sentence boundary label SU at hypothesized word boundary positions. Hence, PMI values can be used to evaluate if a feature is informative. The proposed method is tested with a SUD model built using CRF with training data from a subset of the Gigaword dataset. Experimental results show a reduction of 44.47% in the number of input features while maintaining a F1-score comparable to the baseline model that directly used the original raw input features. This chapter's contributions were published in [22].

The chapter is organized as follows. Section 3.1 describes the baseline system which uses the distant-bigram for feature extraction and CRF for its modeling. Subsequently, a two components of the baseline system are described in Section 3.2 and Section 3.3. The proposed framework is then described in Section 3.4. Experiments are reported in section 3.5. Lastly, section 3.6 discusses the finding and future directions.

3.1 Introduction

SUD task can be viewed as a sequential tagging task to predict a sequence of SU or NSU labels corresponding to an input word sequence. Figure 3.1 depicts the baseline sequential tagging SUD system work-flow including its training and testing phases. During the training phase, the input is a combination of word sequence $S_n = (w_1, \dots, w_i, \dots, w_n)$ and its corresponding target label sequence $L_n = (l_1, \dots, l_i, \dots, l_n)$, where subscript i indicates position in the sequence. Here w_i is a word from a given dictionary D and $l_i \in \{SU, NSU\}$ - where SU and NSU are the ‘‘Sentence Unit’’ or ‘‘Non Sentence Unit’’ labels. Extracted features are represented by $X_n = (x_1, \dots, x_i, \dots, x_n)$ and indexed to the original input word sequence. The label sequence L_n is converted into binary representations, i.e., $Y_n = (y_1, \dots, y_i, \dots, y_n)$, where $y_i \in \{1, 0\}$ and corresponds to SU or NSU. Hence, each pair of an extracted feature and its corresponding label representation (x_i, y_i) at word index i is a training sample for modeling. In the case of a large training dataset is used, the number of training samples are can be huge. The trained model generated during training phase is used for SUD prediction. The output from the testing phase is a sequence of labels NSU or SU representing the SUD prediction result for the input sequence.

Figure 3.1: The baseline SUD system work-flow.

In Figure 3.1, the sequence ‘‘it is such a beautiful day/s should we go out for lunch/s’’ will be used as an example of training input sequence, while the sequence ‘‘how are you

today are you ok” will be used as an example of testing phase input.

3.2 Feature Extraction

The goal of the feature extraction module is to convert the raw input text into a set of features that better represents the data attributes. In this study, we employ distant-bigram [1, 63, 64, 65] for feature extraction. A distant-bigram feature is a word from the left or the right context side, and the word has a distance of k words away from the position of the word boundary being considered. Here $k < 0$ represents the left context and $k > 0$ represents the right context.

Figure 3.2 demonstrates an example of multiple distant-bigram feature extraction for the word boundary at $w_6 = \text{“day”}$ with k ranging from -6 to $+6$ and $k \neq 0$. The set of extracted features is denoted by X_{k6} and each extracted feature is denoted by $x_{k6} = \{w_j, k\}$, where w_j is the word that has distance of j words preceding or following the hypothesized boundary. Specifically, in this example, $j = (6 + (k + 1))$ with $k < 0$ and $j = (6 + k)$ with $k > 0$. For example, with $k = -6$, $x_{-66} = \{w_1 = \text{“it”}, k = -6\}$, and with $k = +2$, $x_{+26} = \{w_8 = \text{“we”}, k = +2\}$.

Note that our baseline system uses multiple distant-bigram features extracted for every word in the input sequence. In which k varies from $-n/2$ to $n/2$ and $k \neq 0$. Here n is a predefined context size of the word boundary being hypothesized. An optimal value for n needs to be tuned for different datasets.

3.3 Conditional Random Fields For SUD Modeling

This work employs CRF [57] as the modeling technique in both the baseline and the proposed frameworks. CRF estimates the conditional probability distribution of label sequence L over the input sequence S . CRF [57] is formulated as follows:

$$P_\lambda(L|S) = \frac{1}{Z(X)} \exp \left(\sum_{i=1}^n \sum_{j=1}^m \lambda_j f_j(y_i, x_i) \right) \quad (3.1)$$

X is a feature set extracted from S , Y is the binary representation of L , n is the length of sequence S and m is the total number of feature functions (details of feature function

Figure 3.2: An example of multiple distant-bigram features for the word boundary at $w_6 = \text{"day"}$ in the sequence. Here $-6 \leq k \leq +6$ and $k \neq 0$.

building is presented shortly), λ_j is a parameter to weight the importance of the feature function $f_j(y_i, x_i)$ to the prediction.

Given a sequence of input feature $X = \{x_1, x_2, \dots, x_n\}$ and its corresponding label sequence $Y = \{y_1, y_2, \dots, y_n\}$, then the input to a CRF modeling is a sequence of feature and label pairs (X, Y) , where:

$$(X, Y) = \{(x_1, y_1), \dots, (x_i, y_i), \dots, (x_n, y_n)\}$$

The process of modeling a CRF model consists of two steps, i.e. feature function building and parameter learning. Firstly, the feature function building generates feature function $f_j(y_i, y_{i-1}, x_i)$ from the given input sequence (X, Y) . Note that $1 \leq j \leq m$ and m is the total number of feature functions that are built from the whole training dataset. Consequently, parameter learning estimates parameter λ_j for each feature function f_j based on the CRF Equation 3.1.

Feature function building: Figure 3.3 demonstrates an example of the feature function building for the extracted features of the word boundary being hypothesized at $w_6 = \text{"day"}$, the extraction process is presented in section 3.2. The input to the CRF Feature Function Building module is twelve distant-bigram features with $-6 \leq k \leq +6$ and $k \neq 0$. The word boundary of w_6 is labeled as $y_j = 1$. The feature function is simply an if-else function representing a guess if the word boundary being hypothesized is labeled as SU ($y_j = 1$) or NSU ($y_j = 0$). A feature function might be assigned a

Figure 3.3: An example of building feature functions for the word boundary being hypothesized at $w_6 = \text{"day"}$ using its twelve distant-bigram features.

real value between from 0 to 1; however, in practice, it is normally set to 1 if its given condition is met; otherwise, it is set to 0. In this example, f_{601} is set to 1 if the distant-bigram $k = -6$ feature x_j is equal to $\{ \text{it}, k = -6 \}$ and its binary label $y_j = 1$, otherwise, it is set to 0. Here the pair (x_j, y_j) represents the feature/label pair that is examined during training.

Given a training dataset, the number of built feature functions is equal to the number number of unique input features $|X|$ multiplied by the number of prediction labels $|L|$, i.e.,:

$$m = |L| \times |X|.$$

Note that in the SUD task, $|L| = 2$.

Parameter Learning: Figure 3.4 demonstrates an example of the parameter learning process. Assume that twelve feature functions are given (as shown at the top of the Figure). During the parameter learning process, for each of the training sequence, a $P(Y|X)$ is obtained using Equation 3.1. There are only a few parameter λ_j ($0 < j \leq m$) that multiplied by a non-zero constants, i.e., only λ_j when its corresponding feature function conditions are matched. For example, when applied to the training sequence

“we have had a good day in a beautiful garden”, only $\lambda_5, \lambda_6, \lambda_7$ corresponding to feature functions f_5, f_6 , and f_7 are multiplied to non-zero, i.e., values of 2, 1, 1 respectively.

Given that the training data contains many labeled sequences, for each training sequence L with its feature set X and the label sequence Y , a $P(Y|X)$ that depends on only λ_j is obtainable. Here $0 < j < m$, where m is the total number of feature functions. To estimate λ_j , a straightforward strategy is used to calculate λ_j base on all $P(Y|X)$. However, in practice, the parameter learning is usually carried by dynamic programming because the number of training sequences is often in the thousands to millions and hence is not feasible for direct calculation.

Figure 3.4: An example of parameter learning.

3.4 Proposed framework

As presented in the previous section, the number of feature functions for a CRF model is equal to the number of m parameters need to be estimated. In the SUD task $m = 2 \times |X|$. The value of m is increased significantly when multiple features are used. Let say we used distant-bigram features from a contextual size of n , i.e. $n/2$ neighbor words from the left and $n/2$ neighbor words from the right of the hypothesized word boundary. The training data size is $|D|$. Then, the number of extracted feature $|X|$ might be up to $n \times |D|$. Note that $|D|$ can be millions of words. Thus, the number of parameters m which is needed to be estimated is huge.

Figure 3.5: The proposed framework with feature selection applying after feature extraction and before modeling.

One way to reduce a model complexity is to reduce the number of input features $|X|$. Reducing $|X|$ can be realized by feature selection techniques. A feature selection method usually uses a relevance measurement between features and target label[66], i.e. $\text{Relevance}(x_i, SU)$, to evaluate if the features is useful to the model building and prediction. In this work, PMI [67] is examined to measure effectiveness and relevance. The proposed framework work-flow is depicted in Figure 3.5. As compared to the baseline, the framework has an additional step, i.e., feature selection, to select only relevant features for modeling. During training phrase, an optimal value for the PMI-based selection threshold will be learned, while during testing phase, the threshold is used to select only informative features for the later prediction.

3.4.1 Pointwise Mutual Information

PMI has been shown to be an effective measurement of relevance between a feature and the target label [66]. It has been a common feature selection method in various machine learning tasks such as selection text categorization [68, 69], sentiment classification [70], computer vision [71] and opinion mining [72].

In this work, PMI measures the relevance score between a distant-bigram feature to the presence of the SU label at the being hypothesized word boundaries. In particular, assuming that x_i is a distant-bigram feature of the word w_i , the PMI score for a feature x_i represents the probability of the word boundary at w_i is labeled as SU, i.e., its binary label $y_i = 1$, given the feature x_i . The PMI formula is shown in Equation 3.2.

$$PMI(x_i, y_i = 1) = \log \frac{p(x_i, y_i = 1)}{p(x_i)p(y_i = 1)} \quad (3.2)$$

Figure 3.6 shows an example of calculating PMI-scores for two distant-bigram features at $k = -6$ and $k = +2$ of the word boundary being hypothesized at $w_6 = \text{“day”}$. Note that we denote the word w_6 as a distant-bigram with $k = -1$ feature to its right boundary.

Figure 3.6: An example of PMI scores obtained for distant-bigram features of the word boundary being hypothesized at $w_6 = \text{“day”}$. The PMI score corresponding to each feature is obtained using Equation 3.2.

3.4.2 PMI-based Feature Selection

PMI-based feature selection is used thresholding technique. Given a selected threshold, a feature is marked as relevant if its PMI score is higher than the threshold. Thus, only features that have PMI scores higher than the threshold are selected for modeling while the low PMI score features are discarded. Figure 3.7 demonstrates the feature selection process for distant-bigram features with $-6 \leq k \leq +6, k \neq 0$ of the word boundary being hypothesized at $w_6 = \text{“day”}$. Here, only features that have PMI scores higher than 0.30 are selected, i.e., the scores highlighted in **bold**.

PMI of distant-k-grams x_{k6} with $k=0.01$	0.04	0.60	0.30	0.45	0.70	0.76	0.25	0.84	0.23	0.46	0.30	
Distant-k-grams x_{k6} with $k =$	-5	-4	-3	-2	-1	+1	+2	+3	+4	+5	+6	
Text sequence S=	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}
	it	is	such	a	beautiful	day	should	we	go	out	for	lunch

Figure 3.7: An example of PMI feature selection for word boundary being hypothesized at w_6 ="day". Here, features are distant-bigram with $-6 \leq k \leq +6$ and $k \neq 0$. Only features that have PMI scores higher than 0.30 are selected, i.e. the scores highlighted in **bold**.

3.4.2.1 Feature Selection For Multiple Distant-bigram Features

When multiple distant-bigram features are employed in a SUD system, defining different selection threshold p_k corresponding to different values of k is necessary. This is because the far context features, e.g., $k < -2$ or $k > 2$, are less dominant to SUD while the near context features, e.g., $-2 \leq k \leq 2$ are more dominant [15]. The multiple distant-bigram feature selection method is illustrated in Figure 3.8. Given an input sequence S with twelve words $w_1, \dots, w_6, \dots, w_{12}$, a multiple set of distant-bigram k x_{ik} for each w_i is obtained. For this input sequence, $1 \leq i \leq 12$ and $-6 \leq k \leq 6$. An input sample is represent by a pair of (x_{ik}, y_{ik}) where the binary label y_{ik} remains unchanged with different k values, it is either 1 or 0. The different values of the predefined p_k are given on the left side in the Figure. With each p_k value, only feature x_{ik} that have $PMI(x_{ik})$ greater than p_k are selected.

Notice that for different k values, an optimal value of selection threshold p_k needs to be forced. The suitable values of p_k is tuned using the development set will be presented in section 3.5.3 in detailed.

3.5 Experiments

This section describes experiments performed on the proposed framework presented in the previous section. The experiments are performed in two sets, the first is to determine various empirical parameters using the development set and the second is to evaluate using the test data. The experiments on the developmental set are to select the appropriate values of the contextual size n , in which $-n/2 \leq k \leq n/2$, and to find the optimal selection threshold p_k for each of distant-bigram features with values of k varying from

Figure 3.8: An example of PMI feature selection applied to multiple distant-bigram features of the entire input sequence. Here, features are distant-bigram from $k = -6$ to $k = +6$. Only features that have PMI scores higher than the corresponding selection threshold p_k are selected. The p_k varies at different values of k .

$-n/2$ to $n/2$. Subsequently, given the selected value of n and p_k , the second experiment evaluates the effectiveness of the proposed feature selection method when tested on an unseen test set.

3.5.1 Data Preparation

Corpus Description. In this work, a subset of English Gigaword 2nd Edition (LDC2005T12)¹ is used. The dataset is released by the Linguistic Data Consortium from The University of Pennsylvania. The complete corpus was archived from several news publishers, e.g., Agence France-Presses, English Service, the New York Times, Newswire Service, over several years. In this work the experiments are conducted on only the news obtained from Agence France-Presses and English Service archived in 1995.

Data Preprocessing. The original corpus is in written text form, thus it has been processed to convert it into a form similar to ASR output. The process includes multiple steps: (1) Removing all digits and special characters; (2) Lowercasing all capitalization;

¹<https://catalog.ldc.upenn.edu/LDC2005T12>

Table 3.1: Number of words and sentences in the dataset after pre-processing. Here, M stands for million, and k stands for thousand.

	# words	# sentences
Gigaiword-Train	28.54M	1.17M
Gigaword-Development	3.50M	144k
Gigaword-Test	3.58M	147k

and (3) Replacing all sentence end representations, i.e., question marks, semi-colons, periods and exclamation marks, by a sentence end label $/s$; and (4) Removing other punctuations, e.g., comma. Subsequently, each word in a paragraph is labeled by either SU or NSU . A word is labeled as SU if it is followed by a $/s$, otherwise it is labeled as NSU . Finally, the dataset is divided into three sets, i.e., the training set for model training, development set for parameter tuning and test set for evaluation. For test set, all sentence labels in the test set are removed. The details about the number of words and sentences per set are shown in Table 3.1.

Dictionary Construction. The dictionary comprises of all the unique words extracted from the corpus. In this work, the size of the dictionary is 80 thousand words.

3.5.2 Experimental Setup

This work uses the following setup in both sets of experiments.

Evaluation Metrics. F1-measure metric.

Features. Distant-bigram k , where $-n/2 \leq k \leq +n/2$, and n is a predefined contextual size.

CRFs Toolkit. CRF++ toolkit² was used to build its CRFs models.

PMI Values. The minimum and maximum PMI values obtained from the given training data are -2 and $+1$ respectively. The PMI value of the word that does not appear within the contextual size of n is set at $-\infty$.

²CRF++: Yet another toolkit: <https://taku910.github.io/crfpp/>

3.5.3 Parameter Tuning

In Section 3.4, we described method to extract and select a single distant-bigram k feature for the baseline and the proposed model. We also noted that our system used multiple distant-bigram features, in which $k \in [-n/2, +n/2]$ and $k \in \mathbb{Z}$. We referred n as the contextual size of the hypothesized word boundary where multiple distant-bigram features are extracted. Further, for applying feature selection on a multiple distant-bigram features, we proposed to use different selection thresholds p_k for different values of k . This section presents our empirical experiments on Gigaword-Development to find out optimal values of n and p_k .

Contextual Size: Figure 3.9 illustrates the F1-scores obtained on the development set using the baseline model when of n increased from 2 to 12. It was seen that the F1-scores improved significantly when n increased from 2 to 8, then slowly saturated when n reached 10. Thus, we choose the context size value $n = 10$ for this specific dataset. The *F1-score* of 58.00% for the development set with $n = 10$, is the the baseline for subsequent experiments. Also, note that a similar experiment on Gigaword-Test with $n = 10$ obtained a *F1-score* of 62.37%.

Figure 3.9: F1-scores (y-axis) obtained when context size n was varied from 2 to 12.

Feature Selection Thresholds: With the selected contextual size of n set at 10, we conducted the second empirical experiment to achieve optimal selection thresholds p_k

for each set of distant-bigram features at k where $-5 \leq k \leq 5$. In this experiment, we used all distant-bigram features within the context n , however, feature selection was only performed on a single distant-bigram position, for example $k = -5$, while all the others with $k \neq 5$ are kept.

Figure 3.10: Feature selection at each single context position within the contextual size of 10.

Figure 3.10 shows the experimental results for the development set. Chart (a) shows results on the left ($k < 0$) context, labeled as L1, L2, L3, L4, L5, where L stands for “Left” and each number represents the relative position of the distant-bigram from the left side of the hypothesized word boundary. Chart (b) shows results on the right side ($k > 0$), labeled as R1, R2, R3, R4, R5, where R stands for “Right” and each number represents the relative position of the distant-bigram from the right side of the hypothesized word boundary. In both (a) and (b), the x-axis represents the PMI thresholds, while the y-axis represents F1-score. The “baseline” label indicates no feature selection was applied. Here, the baseline F1-score is 62.37% as mentioned in the previous experiment.

In most cases, a slight increase was observed in F1-scores when PMI threshold was set to -2 , except L1, L2 and R1, R2. This confirms our observation that the near distant-bigram features, e.g., at the context of L1, L2 and R1, R2 has a higher impact on performance. Specifically, the F1-scores for feature selection at positions L1 and R1 drop dramatically when the PMI thresholds increase, while the F1-scores at position L5, R5 seem to be slightly affected by the increase in selection threshold.

Base on these observations, we only perform multiple distant-bigram feature selection at the context of L5, L4, L3 and R3, R4, R5. All the features at L2, L1, and R1, R2 are

kept. The selected PMI threshold values for the experiment only varied between -2 , -1 and 0 , since it was observed that with a selection threshold higher than 0 , F1-scores significantly decreased in all cases.

3.5.4 Applying Feature Selection

We evaluated the proposed method by performing feature selection concurrently at multiple distant-bigram features with $k = \{-5, -4, -3, +3, +4, +5\}$, that corresponded to the left context L5, L4, L3 and the right context R3, R4, R5. The produced model was denoted as L543R345. The model size and prediction results were compared against the baseline model.

Figure 3.11 shows F1-scores and number of model parameters obtained using both development and test sets when the selection thresholds were varied from baseline (no selection), -2 , -1 and 0 . There was a significant decrease in the number of parameters from 5.36M to 3.12M, a reduction of 44.87% relative to the baseline model, when feature selection was applied with PMI threshold equal to -2 . Interestingly, at this point the F1-score also slightly increased, i.e., 0.2% absolute. At other points, where more features were removed, the number of parameters decreased, but it also resulted in a reduction in F1-scores.

3.6 Conclusion

The experimental results show that PMI-based feature selection method reduces the number of parameters needed to build the CRF model significantly, i.e., a reduction of up to 44.87%. It also shows a slight improvement when the feature selection are at PMI threshold of -2 to multiple distant-bigram features at $k = \{-5, -4, -3, +3, +4, +5\}$. Thus, it can be concluded that PMI is effective for feature selection applied to the SUD task. Additionally, the context size of ten words, i.e., five preceding words and five following words, is optimal for the given dataset. Given the context size of ten surrounding words, the contribution of different distant-bigram features is not equal. The nearer context features, i.e., within two preceding and two following words are dominant, while the far context features, at position 4,5 preceding and following words, are less informative.

Figure 3.11: The F1-scores and number of model parameters obtained on the development and test sets using the feature selection at L543R345 when varying the PMI threshold from “baseline”, -2, -1 and 0.

In this work, we have used the same selection threshold for all distant-bigram features with $k = \{-5, -4, -3, +3, +4, +5\}$. An optimal PMI threshold for each k might further increase system performance. Thus, future work might include a more in-depth investigation into selecting the PMI threshold. Another perspective to explore in the future is the possibility of employing PMI-based feature selection where multiple feature types, such as POS tags, Chunk tags or prosody features, are incorporated. More details on the possible directions for future work will be presented in Chapter 5.

Chapter 4

Investigating Word Embeddings for Deep Learning based SUD

The literature survey presented in Chapter 2 suggested a popular trend for SUD modeling in recent years: Using deep learning. Deep learning is able to automatically learn feature representation, thus it requires less handiwork for feature extraction. Hence, it becomes questionable if refining the input feature space of a deep learning based SUD system still helpful as for the shallow learning system. In this study, we propose a two-stage framework to evaluate the effect of different word embedding variations as input for a deep learning based SUD system. Specifically, the first stage builds different word embedding models using different technologies and databases. Subsequently, at the second stage, each of the embedding models is used to extract word embedding sequences representing the given training sequences for SUD model training using a deep neural network. The resulting SUD models are then evaluated on two different testing scenarios. The first is a same-domain evaluation: The training and test sets have similar characteristics; The second is a domain-mismatch evaluation: The test set has different characteristics from the training set.

The chapter is organized as follows. Section 4.1 briefly describes the proposed framework which consists of two processing stages, i.e., word embedding extraction and SUD model learning. Subsequently, the underlying techniques for the two stages are presented in Section 4.2 and Section 4.3. Experimental setups and results are discussed in Section 4.4. Section 4.5 discusses some related works and experimental results conducted with a written text dataset. Finally, Section 4.6 concludes the chapter.

4.1 Proposed Framework

The proposed framework is illustrated in Figure 4.1. The framework includes two processing stages, Word Embedding Extraction and SUD Model Learning. At the first stage, four embedding models are trained using one of the combinations between the underlying word embedding extraction techniques, i.e. either word-based [24] or subword-based [25, 73], and the training data types, i.e., either speech transcripts (in-domain) or written text (out-of-domain). Each word embedding model is a vocabulary where each entry is a mapping of a word to its dimensional real value vector representation. The second

Figure 4.1: The proposed system architecture

stage perform SUD model learning using a deep neural network consisting of an embedding layer and a combination of biLSTM and fully connected hidden layers. Given a training dataset contains of multiple training samples. Each sample is a word sequence $S = w_1, \dots, w_t, \dots, w_{T-1}, w_T$ with its corresponding label sequence $L = l_1, \dots, l_t, \dots, l_{T-1}, l_T$, in which $l_t \in \{\text{SU}, \text{NSU}\}$. The word sequence S is used as the input to the DNN. At the embedding layer, a real value vector sequence $X = x_1, \dots, x_t, \dots, x_{T-1}, x_T$ that represents S is obtained using one of the embedding models provided by the first stage. Next, X is used as input to the hidden layers for network training, and produces a prediction

of the output label sequence at the output layer $\hat{Y} = \hat{y}_1, \dots, \hat{y}_t, \dots, \hat{y}_{T-1}, \hat{y}_T$, in which $\hat{y}_t \in \{(0, 1), (1, 0)\}$ to represent the prediction label $\hat{l}_t \in \{\text{NSU}, \text{SU}\}$. The prediction layer sequence $\hat{L}$ is then compared against the true label sequence L for network model optimization. The optimization is carried by gradient descent with cross entropy loss function. A more details on practical settings for parameters and hyper-parameters will be described in Section 4.4.2. In the next two sections, the underlying techniques of the two framework stages are discussed.

4.2 Word Embedding Extraction

An embedding vector represents syntax, semantics, and even the morphology of words in context with neighboring words [25, 35]. At the framework first stage, we used two word embedding extraction techniques, i.e. word-based skip-gram embedding proposed by Mikolov et al. [35] and subword-based skip-gram embedding proposed by Bojanowski et al. [25]. This section briefly introduces these two techniques and presents proposed strategies to produce word embedding models.

4.2.1 Word Embedding Types

Word-based Skip-gram Embedding: Word-based skip-gram embedding was proposed by Mikolov et al. [24]. In this technique, the embedding model is a neural network trained to maximize the prediction of the context words of a given word. Assuming an input sequence $\hat{S}_T = w_1, \dots, w_t, \dots, w_T$ is given, where w_t is a word in a dictionary V ; and for each word w_t we want to predict the context $C = w_{t,1}, w_{t,2}, \dots, w_{t,C}$. In this case, the network includes an input layer with size $|V|$, which receives one word vector as the input, a projection layer with size of $|V| \times D$, where D is the number of neurons in the projection layer, and an output layer. The output layer of the network contains $|C|$ distributions, where the i^{th} distribution corresponds to the target context $w_{t,i}$ to be predicted ($1 \leq i \leq |C|$).

The objective function used to train the neural network for the word-based skip-gram embedding is shown in Equation 4.1 [25]:

$$O_T = \sum_{t=1}^T \sum_{c=1}^C \log p(w_c | w_t) \quad (4.1)$$

In which,

$$p(w_c | w_t) = \sum_{w_c \in C} l(s(w_t, w_c)) + \sum_{w_k \in N} l(-s(w_t, w_k)) \quad (4.2)$$

Here, $l : x \rightarrow \log(1 + e^{-x})$ is the logistic function and $s(w_t, w_c)$ is the scoring function between the word w_t and its neighbor w_c and $s(w_t, w_k)$ is the scoring function of the word w_t and its negative samples. The set of negative samples N contains random words outside of the context, i.e. $N \subset V$ and $N \cap C = \emptyset$ [24]. By representing the input word w_t and the output word w_c as vectors u_{w_t} and v_{w_c} respectively, the scoring function is computed by a scalar product between u_{w_t} and v_{w_c} , i.e., $s(w_t, w_c) = u_{w_t}^T v_{w_c}$.

The word-based skip-gram embedding has shown its success in carrying the syntactic and semantic meaning of a word [35]. However, a word vector is formed by its surrounding word-level units without considering the morphological relation within and between words, thus the same vector is used to represent all out-of-vocabulary words.

Subword-based Skip-gram Embedding: Subword-based skip-gram embedding, or dynamic representation of word embedding, is proposed by Bojanowski et al. [25]. The authors used a similar neural network architecture to the word-based skip-gram model. However, their model takes into account not only the word, but also the subword n-grams when calculating a vector representation of a word. For example, the word $w_t = \text{"out"}$ is represented by itself $\langle out \rangle$ and its bi-gram subwords as: $\langle o, ou, ut, t \rangle$ [25].

Assuming G_{w_t} is a set of subword n-grams of w_t , then the scoring function between w_t and a word in its context w_c can be rewritten as shown in Equation 4.3, in which $\mathbf{z}_g$ and $\mathbf{v}_{w_c}$ are n-gram vectors of w_t and the context based word vector is represented by w_c .

$$s(w_t, w_c) = \sum_{g \in G_{w_t}} \mathbf{z}_g^T \mathbf{v}_{w_c} \quad (4.3)$$

The advantage of subword-based skip-gram embedding is its capability to represent rare words in the vocabulary effectively thanks to its representations being shared across words [25]. Additionally, it can represent different out-of-vocabulary words using different vectors [25].

4.2.2 Extraction Strategy

To train word embedding models, we used the above two extraction techniques, and two types of training dataset. The first is “*in-domain*” where the dataset is used both for embedding model training and SUD model training, and the second is “*out-of-domain*” where two separate datasets are used. This strategy results in four word embedding models as listed below:

- (i) The in-domain word-based embedding model.
- (ii) The in-domain subword-based embedding model.
- (iii) The out-of-domain word-based embedding model.
- (iv) The out-of-domain subword-based embedding model.

4.3 SUD Model Learning

The second stage of the proposed framework is model learning performed by a Deep Neural Network (DNN) architecture. DNN is an extended form of the traditional neural network, which is a network of artificial neurons with connecting edges that are organized into layers [74]. A traditional neural network contains three layers, i.e. an input layer, a hidden layer and an output layer. When a neural network has more than one hidden layer, it is a deep neural network. In this study, the proposed framework employs a complex DNN which has an embedding input layer, hidden layers consisting of a mix of fully connected layers and bidirectional LSTM layers, and an output layer with two class distributions corresponding to the two labels SU and NSU. The two types of hidden layers, i.e. fully connected and biLSTM, are briefly described below.

Fully Connected Layer: A neural network layer is a fully connected layer if every neuron in this layer receives input from every neuron in the previous layer. An example of a neural network with all fully connected layers is depicted in Figure 4.2. The network has an input layer, a hidden layer and an output layer. In this example, the neural network performs a two-class classification for the input vector $x = (x_1, x_2, x_3)$, for which the expected output vector is $y = (y_1, y_2)$. The input layer has three-input

neurons representing x_1, x_2 and x_3 . Next, every neuron in the input layer is connected as the input to every neuron in the hidden layer. The number of neurons in the hidden layer can be different from the number of neurons in the input layer. Similarly, every neuron in the hidden layer is connected to every neuron in the output layer. In this example, the output layer has two output neurons, which predicts the two-class distributions, i.e., $y' = (y'_1, y'_2)$, of x over its true label $y = (y_1, y_2)$.

Figure 4.2: A neural network consists of fully connected layers

The computation of a neuron for the above network is as shown in Figure 4.3. The neuron is simply an activation function $f : x \rightarrow y = f(Wx + b)$, which maps from the input $x = (x_1, x_2, x_3)$ to the output y using a weighted matrix $W = [w_1 \ w_2 \ w_3]$ and a bias matrix $b = [b_1 \ b_2 \ b_3]$. The activation function f is normally a nonlinear transformation to ensure variations of the neuron output when it receives different inputs [74].

Figure 4.3: A neuron in a fully connected layer

Fully connected layers learn feature representations from all the features in the previous layer and thus are able to learn the more abstract level of feature representations. However it lacks the ability to carry long-term dependencies [74], and are computationally expensive. Hence, the exclusive use of fully connected layers is usually not suitable for sequential learning tasks [20]. Instead, recent studies suggest that a combined deep neural network with different types of hidden layers can yield better results compared to one-type hidden-layer networks [4, 75].

A Bidirectional LSTM Layer: Bidirectional Recurrent Layers with Long Short-Term Memory [76] cells (biLSTM) are the core component of our model learning architecture. biLSTM is exceptional in handling long-contextual information [76, 77]. A biLSTM layer architecture that produces the new output representation $X' = (x'_1, \dots, x'_{T-1}, x'_T)$ from its input $X = (x_1, \dots, x_{T-1}, x_T)$ is depicted in Figure 4.4. The layer consists of LSTM cells as the hidden states with forward and backward directions. By combining both forward and backward direction, the layer is able to cover both the previous and future context of the current input, which is crucial information for a sequential tagging task likes SUD [20, 27].

Figure 4.4: A single biLSTM layer.

The internal architecture of an LSTM cell for forward direction is depicted in Figure 4.5. The LSTM cell controls information flow through the cell states c_{t-1}, c_t . By performing sigmoid σ , tanh or pointwise multiplication $\otimes$ and add operations, the LSTM

cell is able to allow or prevent historical information to flow through it to the next LSTM cell. Along with the memory gate, the LSTM also produces an output gate h_t which will be the input to the next LSTM cell and also the next layer.

Figure 4.5: The architecture of an LSTM cell [3].

Given the current input x_t , the hidden forward $\vec{h}_t$ state, is calculated using the input from x_t , its previous hidden state $\vec{h}_{t-1}$ and memory cell $\vec{c}_{t-1}$, are as follows[77]:

$$\vec{i}_t = \sigma(\vec{W}_i x_t + \vec{U}_i \vec{h}_{t+1} + \vec{b}_i) \quad (4.4)$$

$$\vec{g}_t = \tanh(\vec{W}_c x_t + \vec{U}_c \vec{h}_{t+1} + \vec{b}_c) \quad (4.5)$$

$$\vec{f}_t = \sigma(\vec{W}_f x_t + \vec{U}_f \vec{h}_{t+1} + \vec{b}_f) \quad (4.6)$$

$$\vec{c}_t = \vec{i}_t \otimes \vec{g}_t + \vec{f}_t \otimes \vec{c}_{t+1} \quad (4.7)$$

$$\vec{o}_t = \sigma(\vec{W}_o x_t + \vec{U}_o \vec{h}_{t+1} + \vec{b}_o) \quad (4.8)$$

$$\vec{h}_t = \vec{o}_t \otimes \tanh(\vec{c}_t) \quad (4.9)$$

Here, σ denotes the sigmoid function, and $\otimes$ denotes the element-wise multiplication. The $\vec{W}_i, \vec{W}_c, \vec{W}_f, \vec{W}_o, \vec{U}_i, \vec{U}_c, \vec{U}_f, \vec{U}_o$ are the weight matrices and $\vec{b}_i, \vec{b}_c, \vec{b}_f, \vec{b}_o$ represent the bias terms.

Similarly, the backward $\overleftarrow{h}_t$ states, based on x_t with its previous hidden state $\overleftarrow{h}_{t+1}$ and memory cell c_{t+1} , are as follows[77]:

$$\overleftarrow{i}_t = \sigma \left(\overleftarrow{W}_i x_t + \overleftarrow{U}_i \overleftarrow{h}_{t+1} + \overleftarrow{b}_i \right) \quad (4.10)$$

$$\overleftarrow{g}_t = \tanh \left(\overleftarrow{W}_c x_t + \overleftarrow{U}_c \overleftarrow{h}_{t+1} + \overleftarrow{b}_c \right) \quad (4.11)$$

$$\overleftarrow{f}_t = \sigma \left(\overleftarrow{W}_f x_t + \overleftarrow{U}_f \overleftarrow{h}_{t+1} + \overleftarrow{b}_f \right) \quad (4.12)$$

$$\overleftarrow{c}_t = \overleftarrow{i}_t \otimes \overleftarrow{g}_t + \overleftarrow{f}_t \otimes \overleftarrow{c}_{t+1} \quad (4.13)$$

$$\overleftarrow{o}_t = \sigma \left(\overleftarrow{W}_o x_t + \overleftarrow{U}_o \overleftarrow{h}_{t+1} + \overleftarrow{b}_o \right) \quad (4.14)$$

$$\overleftarrow{h}_t = \overleftarrow{o}_t \otimes \tanh(\overleftarrow{c}_t) \quad (4.15)$$

The $\overleftarrow{W}_i$, $\overleftarrow{W}_c$, $\overleftarrow{W}_f$, $\overleftarrow{W}_o$, $\overleftarrow{U}_i$, $\overleftarrow{U}_c$, $\overleftarrow{U}_f$, $\overleftarrow{U}_o$ are the weight matrices and the $\overleftarrow{b}_i$, $\overleftarrow{b}_c$, $\overleftarrow{b}_f$, $\overleftarrow{b}_o$ represent the bias terms.

To utilize the advantages of the biLSTM layer and the fully connected layer, the proposed DNN uses both biLSTM and fully connected layers in its hidden layers. The biLSTM layers preserves contextual information, while the fully connected layers extract abstract feature representations. Finally, the highly abstract features are the input to the two-class classification output layer to produce the prediction of a word boundary label which is either an SU or an NSU.

4.4 Experiments

This section describes the experiments conducted on the proposed framework to examines the effectiveness of different embedding models on the performance of the proposed DNN-based SUD model learning that used them as the vocabulary for its embedding layer.

4.4.1 Data and Evaluation Metrics

Dataset for word embedding extraction: As described in section 4.2, two text datasets are used for word embedding extraction, i.e. an in-domain dataset and an out-of-domain dataset. The in-domain dataset consists of reference transcripts obtained from a subset of the MGB2015 Challenge training dataset, namely sMGB2015-Train. The dataset contains multiple-genres of English broadcast news, collected from selected

BBC News channels for several years and have been used as the standard dataset for the MGB 2015 ASR Challenge [78]. The out-of-domain dataset contains first 1 billion (1B) bytes of Wikipedia English text, namely Wiki1B. Both sMGB2015-Train and Wiki1B were pre-processed to remove all punctuation marks and special symbols. Each dataset was converted into just a sequence of words. The number of words and vocabulary size for the two datasets are shown in Table 4.1.

Table 4.1: Dataset used to train word embeddings

Dataset Name	Word Count	Vocabulary Size
sMGB2015-Train	6.62M	241.2k
Wiki1B	124.3M	833,185

Datasets for SUD Model Training and Evaluation: A summary of the datasets used for SUD models training and evaluation is shown in Table 4.2. Here, only the sMGB2015-Train dataset is used to train SUD models, while the others are used for evaluation. The first evaluation set, namely sMGB2015-Eval, contains reference transcripts obtained from a subset of the development set of the MGB2015 dataset. As the same genre transcripts from sMGB2015-Train are used for SUD models training, the evaluation results reported on the sMGB2015-Eval is considered as the same-domain testing condition. Apart from that, to test the generalization of the hypothesis, we used reference transcripts and ASR transcripts of two hours of RT04F¹ as the test sets for the domain-mismatch condition, namely RT04F-REF and RT04F-ASR respectively. RT04F-ASR transcripts were obtained using our in-house ASR system with WER at 29.5%. The sentence boundaries for ASR transcripts was obtained by time boundary alignment between the ASR transcripts and REF transcripts.

Evaluation Metrics: This work uses standard Precision, Recall and F1-score as the evaluation metrics. The metrics are defined as follows:

$$Precision = TP / (TP + FP)$$

$$Recall = TP / (TP + FN)$$

¹<https://catalog.ldc.upenn.edu/LDC2005T24>

Table 4.2: Datasets used for SUD model training and evaluation

Dataset	Word Count	Sentence Count	Average Sentence Length
sMGB2015-Train	6.62M	793,966	8.43
sMGB2015-Eval	138.77k	18.41k	7.54
RT0304-REF	13.97	1,162	12.02
RT0304-ASR	15.10	1,162	13.00

Table 4.3: The confusion matrix between the system output and reference.

	System True	System False
Reference True	TP	FN
Reference False	FP	TN

$$F1 - score = \frac{2 * Precision * Recall}{Recall + Precision}$$

Where TP (True Positive), FN (False Negative) and FP (False Positive) are calculated based on the confusion matrix in Table 4.3. Note that the total number of word boundaries is $TP + FN + FP + TN$ and the total number of true sentence unit boundaries is $TP + FN$.

4.4.2 Experimental Setup

Parameters for word embedding extraction: We used word2vec [24, 35] to train and generate word models for the word-based skip-gram embedding. To train and generate subword-based models, we used fastText [25, 73, 79]. For sub-word n-gram, we chose n varies from 3 to 6. For both word2vec and fastText, we set minimum word count to 1. The default settings provided by the toolkits were used for the other parameters.

Hyper-parameters for the DNN: The following network setting and hyper-parameters were heuristically chosen for the SUD model learning stage. The proposed DNN architecture consists of an embedding layer, two biLSTM hidden layers, three fully connected hidden layers and an output layer. The embedding layer has 100 neurons to

extract a 100 dimensional word vector per input word. Each of the biLSTM layers consists of 100 hidden states and each of the fully connected layers has 128 hidden states. The output layer produces two-class distributions of SU or NSU labels. Furthermore, we also used a dropout of 0.5 at the embedding layer, the biLSTM layers and the last fully connected hidden layer. Cross-entropy is used as the loss function and network optimization was performed using gradient descent with Adam optimizer. The maximum training epochs was set at 500, with early stop in the event that there were no further improvements over 5 continuous epochs.

SUD Model Learning: The second stage of the proposed framework perform SUD model learning using a SUD training dataset and the word embedding models produced in the first stage. Specifically, given the training dataset is sMGB2015-Train, we separately use one of the four word embedding models to extract word embedding sequence representing the original training sequence at the embedding layer. Subsequently, the word embedding sequences are used to train the DNN and finally produces an SUD model. As a result, four SUD models are produced corresponding to the four word embedding models:

- (i) The SUD model trained using the in-domain word-based embedding model and the sMGB2015-Train - **mSkip_MGB**
- (ii) The SUD trained using the in-domain subword-based embedding model and the sMGB2015-Train- **msSkip_MGB**
- (iii) The SUD model trained using the out-of-domain word-based embedding model and the sMGB2015-Train - **wSkip_MGB**
- (iv) The SUD model trained using the out-of-domain subword-based embedding model and the sMGB2015-Train - **wsSkip_MGB**

4.4.3 Evaluation Results

We conducted two evaluation types, i.e., same-domain evaluation and domain-mismatch evaluation, on the four trained SUD models as described in the previous section. The same-domain evaluation reports experimental results on the same-domain evaluation

dataset sMGB2015-Eval which has the same genres as the training set sMGB2015-Train. The domain-mismatch evaluation shows results when tested the SUD models with different genres datasets, i.e., RT04-REF and RT04-ASR. The detailed experimental results are reported below.

Same-domain Evaluation: Table 4.4 shows the evaluation results when the above four models were tested with MGB2015-Eval. It can be seen that the in-domain sub-word based embedding model, **msSkip_MGB**, yielded the best result compared to the others. Surprisingly, the SUD model with out-of-domain word-based embedding **wSkip_MGB** achieved the second highest F1-score, which was only 0.45% lower than the **wsSkip_MGB** model. Another point to note is that in case of using out-of-domain embedding models, SUD model with word-based embedding **wSkip_MGB** is marginally better than the SUD model with subword-based embedding **wSkip_MGB** model, which contrasts to the case of using the in-domain embeddings.

Table 4.4: Experimental results for same-domain evaluation

Model name	sMGB2015-Eval		
	P	R	F1
mSkip_MGB	83.39	66.70	74.33
msSkip_MGB	82.53	71.13	76.40
wSkip_MGB	81.97	77.07	75.95
wsSkip_MGB	83.07	68.21	74.91

Domain-Mismatch Evaluation: Table 4.5 presents the results of the domain-mismatch evaluation for the above four SUD models. The models were tested with both reference and ASR transcripts. It is observed that overall performance was much lower than the previous same-domain condition evaluation results. We also observed the same behavior with the same-domain testing condition, where the sub-word based models showed better results when trained on in-domain data while the word-based models showed better results with the large out-of-domain dataset. Interestingly, **msSkip_MGB** gave approximately 2% and 1% improvement in F1-score compared to **wSkip_MGB** when tested on the RT04F reference and ASR transcription respectively.

Table 4.5: Experimental results for the domain-mismatch evaluation

Model name	RT04F-REF			RT04F-ASR		
	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>
mSkip_MGB	79.68	39.16	52.51	76.11	35.40	48.32
msSkip_MGB	77.65	46.64	58.28	72.76	41.17	52.58
wSkip_MGB	76.13	44.75	56.37	71.39	40.40	51.60
wsSkip_MGB	76.98	43.46	55.56	73.35	38.93	50.90

4.5 Discussion

4.5.1 Related Work

Word embedding representations have shown to be effective in carrying the syntax, semantics [24, 35] and even morphology [25] of words. Many word embedding variations have been used in the state-of-the-art SUD models. For example, Che et al. [17] used character-based embedding, Xu et al. [27] used unsupervised word-based embedding and Xu et al. [20] used supervised word-based embedding. However, there is limited efforts on examining the effectiveness of different word embeddings to the SUD task.

Two similar efforts to our work have been done recently. The work done by Treviso et al. [80] compares various pre-trained word-based and sub-word based embeddings for SUD for Brazilian Portuguese learned by CNN-RNN architecture. The tested embeddings were trained on a combined out-of-domain text dataset, and their dimension ranged from fifty to six hundred. The experimental results showed that none of the tested embeddings had clear improvement compared to the others. Another work by Li et al. [81] also compared various pre-trained word embeddings for the text segmentation task using a pointer network. However, the work carried only a comparison between the proposed model self-learning word embedding, GloVe² and two other public available pre-trained embeddings, Google Embeddings³ and FastText Embeddings⁴.

²<http://nlp.stanford.edu/projects/glove/>³<https://code.google.com/archive/p/word2vec/>⁴<https://fasttext.cc/docs/en/english-vectors.html>

4.5.2 Another generalization test on a written dataset

We conducted a similar experiment as presented in Section 4.4 on a written dataset to test the the proposed framework generalization. Specifically, we replaced the roles of sMGB2015-Train and sMGB2015-Eval by Gigaword-Train and Gigaword-Test, which are also the training and test sets used in Chapter 3. Gigaword-Train is used both for training the in-domain word embedding models and the SUD models, while Gigaword-Test is the same-domain test set. The produced SUD models are annotated as follows:

- (i) The SUD model trained using in-domain word-based embedding with Gigaword-Train - **gSkip_GW**
- (ii) The SUD model trained using in-domain word-based embedding with sGigaword-Train - **gsSkip_GW**
- (iii) The SUD model trained using out-of-domain word-based embedding with Gigaword-Train - **wSkip_GW**
- (iv) The SUD model trained using out-of-domain word-based embedding with Gigaword-Train - **wsSkip_GW**

In this experiment the in-domain word-based and subword-based embedding models are trained using Gigaword-Train, thus they are named differently, i.e. **gSkip** and **gsSkip** for the word-based and the subword-based models respectively. On the other hand, there is no change on the out-of-domain word-based **wSkip** and subword-based **wsSkip** models.

Table 4.6 shows experimental results obtained on both the same-domain test set, i.e., Gigaword-Test, and the two domain-mismatch test sets, i.e. using RT04-REF and RT04-ASR. In this scenario, the results share two similar observations to the previous experiment: Firstly, the in-domain embedding model achieves the best results on all the test sets; Secondly, the out-of-domain word-based word embedding is better than subword-based embedding. However, notice that in case the in-domain dataset is used, it is not clear if the word-based or subword-based embedding is better, as the obtained F1 scores on the same-domain test set show only 0.13% different.

It is also important to note that the F1 scores obtained on RT04-REF and RT04F-ASR in this experiment are much lower than the previous corresponding SUD models trained with sMGB2015-Train. For example **wsSkip_MGB** achieves 56.37% and 52.58% of F1 scores while the **wsSkip_GW** achieves only 47.61% and 41.98% when tested on RT04F-REF and RT04F-ASR respectively. The results are explainable as sMGB2015-Train contains broadcast news transcripts, which are much more similar to RT04-REF and RT04-ASR as compared to the written text from Gigaword-Train.

Table 4.6: Evaluation results for the SUD models trained using Gigaword-Train, tested on both same-domain and domain-mismatch conditions

Model name	Same-domain			Domain-mismatch					
	Gigaword-Test			RT04F-REF			RT04F-ASR		
	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>	<i>P</i>	<i>R</i>	<i>F1</i>
gSkip_GW	79.85	66.74	72.71	83.23	35.46	49.73	75.76	32.30	45.29
gsSkip_GW	79.31	66.90	72.58	81.11	35.11	49.00	71.82	31.61	43.90
wSkip_GW	80.32	64.31	71.43	85.14	33.05	47.61	74.67	29.20	41.98
wsSkip_GW	78.35	64.17	70.56	84.63	32.70	47.18	76.00	29.20	42.19

4.6 Conclusion

The study provided a comprehensive comparison on the effectiveness of different word embedding models to SUD for speech transcripts using deep learning based model learning method. Specifically, a two-stage framework consisting of word embedding extraction and deep learning based SUD model learning has been proposed. The word embedding stage allows to train multiple word embedding models with different settings. At this stage, four word embedding models are obtained by varying the choices of word embedding types, i.e., word-based embedding [35] and subword-based embedding [25], and the choices of word embedding training databases, i.e., in-domain and out-of-domain. Subsequently, each of the four word embedding model is used to extract word embedding sequences corresponding to the training word sequences for the second stage of SUD model learning. At this stage, the same DNN architecture and speech transcript database

are used to train the SUD models, the only difference is the choice of the word embedding models. Experimental results showed that the subword-based word embedding trained using the in-domain data gains the best F1-score in overall. On the other hand, the word-based embedding model works better than the subword-based model when the out-of-domain data is used to train the word embeddings. Additionally, it can be seen that even a SUD model trained with a large dataset consisting of approximately 6.62 million word transcripts, it still suffers significantly from the domain-mismatch condition.

Chapter 5

Conclusions

This thesis approaches to optimize lexical features as input for SUD systems. Two approaches were proposed, the first approach focused on optimizing distant-bigram features for shallow learning based systems, i.e., CRF based SUD system. The second approach focused on analyzing the effectiveness of different word embedding types as input for deep learning based SUD systems. The details of the contributions are summarized in the following section and the future works are discussed in the later section.

5.1 Contributions

5.1.1 Pointwise Mutual Information for feature selection

The first contribution is about the use of PMI as a correlation measurement for feature selection. This work has been published in the 8th ACIIDS 2016 Conference [22] and discussed in Chapter 3. This work studied the use of distant-bigram as input feature for a CRF based SUD model. Experimental results showed that the contribution of distant- k -gram features to the prediction of SU labels depended on the value of k . The nearer context distance- k -gram features, i.e. $|k| \leq 2$, are more dominant than the farther ones, i.e. $|k| > 2$. Furthermore, a work by Zhang [66] demonstrated the success of using PMI as a feature selection method to filter out non-dominant features in the input feature spaces. Based on these two observations, we proposed to use PMI as the metric to measure the correlation of a distant-bigram feature to the prediction of the SU label, so as to retain only dominant features for the next step of SUD modeling. Our results also showed that the proposed PMI-based feature selection significantly reduced the input feature space,

thus reducing the size of trained models by up to 44.87% while maintaining a comparable performance to the original model. In conclusion, we found that a careful investigation into selecting suitable selection thresholds, from PMI values might improve performance further.

5.1.2 Study of word embedding types

The second contribution is a study to explore the optimization of the lexical input feature space. The study examined various word embedding types, i.e. word-based embedding and subword-based embedding, as an input for a deep learning based SUD model. Our paper on the proposed approach has been accepted by the IALP 2018 Conference [23]. Our experimental results suggested that subword-based embedding trained with the in-domain dataset is the best choice among all the word embedding variations examined, while word-based embedding showed a better result when trained with the out-of-domain dataset. Furthermore, the study showed a significant decrease in F1-scores when the trained SUD model was tested on domain-mismatch conditions, although the model had been trained with a large dataset. System performance was also affected when the testing condition was switched from reference transcripts to ASR transcripts.

The outcomes of this study can also be used as benchmarks for future studies on SUD systems that employ word embeddings and deep learning techniques. Depending on the available training datasets, either in-domain or out-of-domain, the choice of word embedding types can be adjusted accordingly to achieve the best result.

5.2 Future Work

Future work can focus further on techniques to the two proposed approaches presented in Chapter 3 and Chapter 4. A combination of the two proposed approaches might be taken into consideration as it benefits from the advantages of both approaches in improving SUD systems in both efficiency and performance.

Further investigation into PMI-based feature selection. In Chapter 3, the proposed PMI-based feature selection method was presented. In the experiments, a single value was applied as a selection threshold for all contextual features. Even though

using a global threshold has proved to be helpful, customizing separate thresholds for each contextual features might improve performance. For example, using lower thresholds to retain more of the nearer context features and using higher thresholds to effectively prune the farther ones. This may further improve system performance and reduce the input feature space.

Additionally, experiments were performed on only lexical features that were derived from text data source. There was also an assumption of independence between these features. Thus, there is still a need to explore the applicability of the method for other features such as POS tags, or Chunk tags. Consequently, the effectiveness of the method when applied to a combination of multiple features, especially those which are highly correlated, is still undiscovered.

Further investigation into word embeddings and deep learning techniques.

Our work, presented in Chapter 4, showed that subword-based embedding obtained the best result when the in-domain dataset is available for word embedding training. On the other hand, a recent work by Xu et al. [20] showed that supervised word-based embedding achieved a better result as compared to unsupervised word-based embedding when used as input for a biLSTM SUD model. This suggests a potential direction of extracting supervised subword-based word embedding for SUD. Another potential direction is to improve the deep learning architecture for model learning. Our work employed a combination of biLSTM and fully connected layers as the core components for the DNN. This was to take advantage of the two layer’s capabilities, i.e. a biLSTM layer is able to capture contextual information while a fully connected layer can learn abstract feature more efficiently. Another recent work by Can et al. [82] showed that the CNN architecture is useful in learning local information for SUD, which might not be captured by our current DNN architecture. This opens the question of how to use the CNN architecture in SUD, by partially or fully replacing the current DNN architecture by the CNN, or by adding the CNN as a part of the current DNN architecture.

Using PMI scores in deep learning based SUD model. Lastly, a combination of the above techniques completes the picture of our future work, both in features and modeling practices. Specifically, there is a potential to use PMI values for the mixed DNN model. These values might act as a selection criteria for feature selection (similar

to the work proposed in Chapter 3), or as an additional feature to further emphasize those features which are highly correlated with the presence or absence of SU labels.

References

- [1] T. Y. Chong, “Exploiting long context using distance and occurrence information for language modeling.” Ph.D. dissertation, Nanyang Technological University, 2017.
- [2] F. Pausch, “Automatic segmentation of speech into sentences using prosodic features,” *Master’s Thesis submitted to the Institute of Electronic Music and Acoustics of the University of Music and Performing Arts Graz*, November 2011.
- [3] C. Olah, “Understanding lstm networks,” 2015. [Online]. Available: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [5] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, “Deep speech 2: End-to-end speech recognition in English and Mandarin,” in *International Conference on Machine Learning*, 2016, pp. 173–182.
- [6] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, “The Microsoft 2016 conversational speech recognition system,” in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5255–5259.
- [7] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens *et al.*, “Moses: Open source toolkit for

REFERENCES

- statistical machine translation,” in *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, 2007, pp. 177–180.
- [8] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [9] E. M. Voorhees and D. M. Tice, “The trec-8 question answering track evaluation.” in *TREC*, vol. 1999, 1999, p. 82.
- [10] H. Yu and V. Hatzivassiloglou, “Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences,” in *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, 2003, pp. 129–136.
- [11] R. Speer and C. Havasi, “Conceptnet 5: A large semantic network for relational knowledge,” in *The Peoples Web Meets NLP*. Springer, 2013, pp. 161–176.
- [12] D. Rajagopal, E. Cambria, D. Olsher, and K. Kwok, “A graph-based approach to commonsense concept extraction and semantic similarity detection,” in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 565–570.
- [13] Y. Liu, E. Shriberg, A. Stolcke, , D. Hillard, M. Ostendorf, and M. Happer, “Enriching speech recognition with automatic detection of sentence boundaries and disfluencies,” *Audio, Speech, and Language Processing, IEEE Transactions*, vol. 14, no. 5, pp. 1526–1540, 2006.
- [14] J. Kolář, “Automatic segmentation speech into sentence-like units.” Ph.D. dissertation, University of West Bohemia in Pilsen, Pilsen, Czech, 2008.
- [15] A. Gravano, M. Jansche, and B. Michiel, “Restoring punctuation and capitalization in transcribed speech,” in *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2009.

REFERENCES

- [16] B. Favre, R. Grishman, D. Hillard, H. Ji, D. Hakkani-Tür, and M. Ostendorf, “Punctuation speech for information extraction,” in *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008.
- [17] X. Che, C. Wang, H. Yang, and C. Meinel, “Punctuation prediction for unsegmented transcript based on word vector.” in *Proceedings of the Language Resources and Evaluation Conference (LREC)*, 2016.
- [18] A. Stolcke and E. Shriberg, “Automatic linguistic segmentation of conversational speech,” in *Fourth International Conference Spoken Language Processing (ICSLP)*, vol. 2, 1996, pp. 1005–1008 vol.2.
- [19] Y. Liu, “Structural event detection for rich transcription of speech.” Ph.D. dissertation, Purdue University, 2004.
- [20] C. Xu, L. Xie, and X. Xiao, “A Bidirectional LSTM approach with word embeddings for sentence boundary detection,” *Journal of Signal Processing Systems*, pp. 1–13, 2017.
- [21] O. Klejch, P. Bell, and S. Renals, “Punctuated transcription of multi-genre broadcasts using acoustic and lexical approaches,” in *Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 433–440.
- [22] T. N. Ho, T. Y. Chong, V. H. Do, V. T. Pham, and E. S. Chng, “Improving efficiency of sentence boundary detection by feature selection,” in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2016, pp. 594–603.
- [23] T. N. Ho, D. C. Can, and E. S. Chng, “An investigation of word embeddings with deep bidirectional lstm for sentence unit detection in automatic speech transcription,” in *The 2018 International Conference on Asian Language Processing*. IEEE Computer Society, accepted.
- [24] T. Mikolov *et al.*, “Distributed representations of words and phrases and their compositionality,” in *NIPS*, 2013, pp. 3111–3119.

REFERENCES

- [25] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *arXiv preprint arXiv:1607.04606*, 2016.
- [26] C. Xu, L. Xie, G. Huang, X. Xiong, E. S. Chng, and H. Li, “A deep neural network approach for sentence boundary detection in Broadcast News.” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2014.
- [27] K. Xu, L. Xie, and K. Yao, “Investigating LSTM for punctuation prediction,” in *Chinese Spoken Language Processing (ISCSLP), 2016 10th International Symposium*. IEEE, 2016, pp. 1–5.
- [28] W. Lu and H. T. Ng, “Better punctuation prediction with dynamic conditional random fields,” in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010, pp. 177–186.
- [29] Y. Gotoh and S. Renals, “Sentence boundary detection in broadcast speech transcripts,” in *ISCA Workshop: Automatic Speech Recognition: Challenges for the New Millennium ASR-2000*, 2000.
- [30] M. Paul, “Overview of the IWSLT 2009 evaluation campaign,” in *International Workshop on Spoken Language Translation (IWSLT) 2009*, 2009.
- [31] A. S. E. Stolcke, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tür, and Y. Lu, “Automatic detection of sentence boundaries and disfluencies based on recognized word.” in *Proceeding of ICSLP98, Sydney, Australia*, 1998.
- [32] S. Andreas, *SRILM - An extensible language model toolkit*, SRI International, <http://www.speech.sri.com/projects/srilm/>, 2002.
- [33] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Otendorf, B. Peskin, and M. Harper, “Comparing and combining generative and posterior probability models: Some advances in sentence boundary detection in speech,” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2004.

REFERENCES

- [34] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.
- [35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [36] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.
- [37] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron *et al.*, “Theano: Deep learning on gpus with python,” in *NIPS 2011, BigLearning Workshop, Granada, Spain*, vol. 3. Citeseer, 2011.
- [38] *NIST Rich Transcription 2003 Fall Evaluation Plan Presentation (RT03F)*, National Institute of Standards and Technology (NIST), <http://www.nist.gov/speech/tests/rt/rt2003/fall/presentations/>, 2003.
- [39] G. Huang, C. Xu, L. Xie, E. S. Chng, and H. Li, “Multi-view features in a DNN-CRF model for improved sentence unit detection on English broadcast news.” in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference 2014*, 2014.
- [40] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural Language Processing (almost) from scratch,” *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.
- [41] R. Collobert, “Deep learning for efficient discriminative parsing,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 224–232.
- [42] K. Toutanova and C. D. Manning, “Enriching the knowledge sources used in a maximum entropy part-of-speech tagger,” in *Proceedings of the 2000 Joint SIG-DAT conference on Empirical methods in natural language processing and very large*

- corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*. Association for Computational Linguistics, 2000, pp. 63–70.
- [43] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, “Feature-rich part-of-speech tagging with a cyclic dependency network,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 2003, pp. 173–180.
- [44] J. Huang and G. Zweig, “Maximum Entropy Model For Punctuation Annotation From Speech,” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2002.
- [45] Y. Liu, A. Stolcke, E. Shriberg, and M. Harper, “Using conditional random fields for sentence boundary detection in speech,” in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL05)*, 2005, pp. 451–458.
- [46] M. Ballesteros and L. Wanner, “A neural network architecture for multilingual punctuation generation,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1048–1053.
- [47] O. Tilk and T. Alumäe, “LSTM for punctuation restoration in speech transcripts,” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Dresden, Germany, 2015.
- [48] D. Wang and S. Narayana, “A multi-pass linear fold algorithm for sentence boundary detection using prosodic cues.” in *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.
- [49] E. Shriberg, A. Stolcke, D. Hakkani-Tür, and G. Tür, “Prosody-based automatic segmentation of speech into sentences and topics,” *Speech communication*, vol. 32, no. 1-2, pp. 127–154, 2000.
- [50] K. Hadding-Koch, “Systems of prosodic and paralinguistic features in english,” 1967.

REFERENCES

- [51] J. Vaissière, “Language-independent prosodic features,” in *Prosody: Models and Measurements*. Springer Verlag, 1983.
- [52] Z. Huang, L. Chen, and M. Harper, “An open source prosodic feature extraction tool,” in *Proceedings of the Language Resources and Evaluation Conference (LREC)*, 2006.
- [53] D. Wang, L. Lu, and H.-J. Zhang, “Speech segmentation without speech recognition,” in *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.
- [54] B. Favre, D. Hakkani-Tür, and E. Shriberg, “Syntactically-informed models for comma prediction,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2009, pp. 4697–4700.
- [55] J. Read, R. Drizan, S. Oepen, and L. J. Solberg, “Sentence boundary detection: A long solved problem?” *Proceedings of COLING 2012: Posters*, pp. 985–994, 2012.
- [56] H. Christensen, Y. Gotoh, and S. Renals, “Punctuation annotation using statistical prosody models,” in *ISCA tutorial and research workshop (ITRW) on prosody in speech recognition and understanding*, 2001.
- [57] J. Lafferty, A. McCallum, and F. C. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data.” *Proceeding of International Conference on Machine Learning*, 2001.
- [58] F. Sha and F. Pereira, “Shallow parsing with conditional random fields,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 2003, pp. 134–141.
- [59] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

REFERENCES

- [60] W. Gale and S. Parthasarathy, “Experiments in character-level neural network models for punctuation,” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2017, pp. 2794–2798.
- [61] S. Cuendet, D. Hakkani-Tür, and G. Tur, “Model adaptation for sentence segmentation from speech.” Spoken Language Technology Workshop, 2006.
- [62] U. Guz, S. Cuendet, D. Hakkani-Tür, and G. Tur, “Co-training using prosodic and lexical information for sentence segmentation,” in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2007.
- [63] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld, “The sphinx-ii speech recognition system: an overview,” *Computer Speech & Language*, vol. 7, no. 2, pp. 137–148, 1993.
- [64] M. Simons, H. Ney, and S. Martin, “Distant bigram language modelling using maximum entropy,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1997, p. 787.
- [65] D. Klakow, “Log-linear interpolation of language models,” in *Fifth International Conference on Spoken Language Processing*, 1998.
- [66] Y. Zhang, “Grouping feature in big dimensionality.” Ph.D. dissertation, Nanyang Technological University, 2016.
- [67] G. Bouma, “Normalized (pointwise) mutual information in collocation extraction,” *Proceedings of GSCL*, pp. 31–40, 2009.
- [68] K.-M. Schneider, “Weighted average pointwise mutual information for feature selection in text categorization,” in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2005, pp. 252–263.
- [69] Y. Xu, G. J. Jones, J. Li, B. Wang, and C. Sun, “A study on mutual information-based feature selection for text categorization,” *Journal of Computational Information Systems*, vol. 3, no. 3, pp. 1007–1012, 2007.

REFERENCES

- [70] A. Abbasi, H. Chen, and A. Salem, “Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums,” *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, p. 12, 2008.
- [71] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, “Crisp boundary detection using pointwise mutual information,” in *European Conference on Computer Vision*. Springer, 2014, pp. 799–814.
- [72] Q. Su, K. Xiang, H. Wang, B. Sun, and S. Yu, “Using pointwise mutual information to identify implicit features in customer reviews,” in *International Conference on Computer Processing of Oriental Languages*. Springer, 2006, pp. 22–30.
- [73] A. Joulin *et al.*, “Fasttext. zip: Compressing text classification models,” *arXiv preprint arXiv:1612.03651*, 2016.
- [74] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [75] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.
- [76] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [77] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.
- [78] P. Bell *et al.*, “The mgb challenge: Evaluating multi-genre broadcast media recognition,” in *ASRU*. IEEE, 2015, pp. 687–693.
- [79] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” *arXiv preprint arXiv:1607.01759*, 2016.

REFERENCES

- [80] M. V. Treviso *et al.*, “Evaluating word embeddings for sentence boundary detection in speech transcripts,” *arXiv preprint arXiv:1708.04704*, 2017.
- [81] J. Li, A. Sun, and S. Joty, “Segbot: A generic neural text segmentation model with pointer network,” in *Proceedings of 27th IJCAI, July 13-19, 2018, Stockholm, Sweden.*, 2018, pp. 4166–4172.
- [82] D. C. Can, T. N. Ho, and E. S. Chng, “A hybrid deep learning architecture for sentence unit detection,” in *The 2018 International Conference on Asian Language Processing*. IEEE Computer Society, accepted.