IDIAP RESEARCH REPORT

Model Adaptation for Sentence Unit Segmentation from Speech

Sébastien Cuendet [a] [b]

IDIAP–RR 06-64

October 2006

PUBLISHED IN
EPFL Masters Thesis

[a] L'IDIAP Laboratory, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland
[b] Speech Group, International Computer Science Institute, Berkeley, CA, USA

IDIAP Research Institute **www.idiap.ch**
Rue du Simplon 4 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 06-64

Model Adaptation for Sentence Unit Segmentation from Speech

Sébastien Cuendet

October 2006

published in
EPFL Masters Thesis

Abstract

The sentence segmentation task is a classification task that aims at inserting sentence boundaries in a sequence of words. One of the applications of sentence segmentation is to detect the sentence boundaries in the sequence of words that is output by an automatic speech recognition system (ASR). The purpose of correctly finding the sentence boundaries in ASR transcriptions is to make it possible to use further processing tasks, such as automatic summarization, machine translation, and information extraction.

Being a classification task, sentence segmentation requires training data. To reduce the labor-intensive labeling task, available labeled data can be used to train the classifier. The high variability of speech among the various speech styles makes it inefficient to use the classifier from one speech style (designated as out-of-domain) to detect sentence boundaries on another speech style (in-domain) and thus, makes it necessary for one classifier to be adapted before it is used on another speech style.

In this work, we first justify the need for adapting data among the broadcast news, conversational telephone and meeting speech styles. We then propose methods to adapt sentence segmentation models trained on conversational telephone speech to meeting conversations style. Our results show that using the model adapted from the telephone conversations, instead of the model trained only on meetings conversation style, significantly improves the performance of the sentence segmentation. Moreover, this improvement holds independently from the amount of in-domain data used.

In addition, we also study the differences between speech styles, with statistical measures and by examining the performances of various subsets of features. Focusing on broadcast news and meeting speech style, we show that on the meeting speech style, lexical features are more correlated with the sentence boundaries than the prosodic features, whereas it is the contrary on the broadcast news. Furthermore, we observe that prosodic features are more independent from the speech style than lexical features.

Keywords: sentence segmentation from speech, model adaptation, meta-data extraction, boosting, prosodic and lexical features.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor at ICSI, Dilek Hakkani-Tür, for always keeping her door open and taking the time to answer my technical and life-related questions. Working with her has been instructive and a great experience.

I am also thankful to Prof. Hervé Bourlard for his trust and for giving me the opportunity to discover Berkeley and California.

I have to mention my long-time close friend, classmate and colleague Jean-Philippe "Short One" Pellet and thank him for the fertile, late-night IM debates between Zurich and Berkeley on machine learning and the meaning of the derivative of the life–value function, as well as for his precious help with LaTeX.

Thank you to all the persons that have helped me finalize this work during the last weeks. I am in particularly indebted to Mathew Magamai who patiently initiated me to the scientifical style of writing and to all the friends who generously devoted some of their time to give me linguistic advice.

Finally, I would like to thank every member of the Speech Group at ICSI for their hearty welcome into the group and their openness to discussion. Working – and playing Foosball! – with such people has been a real pleasure.

This work was supported by the Swiss National Science Foundation through the research network IM2.

CHAPTER 1

Introduction

Throughout the last decades, tasks related to the speech processing domain have gained importance. One of the reasons for this is the increase of the computer abilities, in terms of processing speed and memory size, and the rapid development of the world wide web, all of which have made it possible to share, store and process audio and video data. Another reason for this growth of interest toward speech related tasks is the huge progress that has been made in speech recognition since the beginning of the development of speech recognition systems in the 1930's. Since then, lots of research has been conducted in the speech recognition field and the improving results have opened new horizons in the exploitation of speech. For example, applications have been developed that allow the user to interact with a system by speaking to it (Jurafsky, 1994; Gorin *et al.*, 1997, among others).

One of these new horizons is the retrieval of the information contained in speech data. Indeed, the progress of the automatic speech recognition systems (ASR) have led to more accurate textual transcriptions of speech, allowing researchers to apply textual processing techniques (question answering, summarization, etc.) to speech transcriptions. However, two main problems have appeared in the processing of speech transcriptions. First, despite the progress in the automatic transcription of speech, the word error rate of nowadays ASR is still high and the information retrieval techniques are therefore not as efficient as on textual data. Second, the sequence of words output by ASR misses information that are inherent in textual data. To illustrate this lack of information, an example of ASR output is given in Figure 1.1, and the same sequence of words with speaker boundaries, punctuation and capitalization is shown in Figure 1.3, while the correct sequence of words is shown in Figure 1.2. A comparison of Figure 1.1 and Figure 1.3 shows that the ASR output of Figure 1.1 in particular lacks capitalization, marks of speaker turn change and punctuation.

These informations are assumed by the textual tasks mentioned above. For example, in the automatic summarization task, one approach is to form a summary by concatenating the sentences from the original text, that contain the most information (Mrozinski *et al.*, 2005). To extract sentences from the original text, the system has to know where the sentence boundaries are. Thus, when done on speech transcriptions, automatic summarization needs the sentences to be detected first by a sentence segmentation classifier. Adding information to the ASR output, in this case the sentence boundaries, is part of a meta-data extraction process that aims at enriching the ASR output to guide further textual tasks.

Sentence segmentation is part of this meta-data extraction, or enrichment process together with

```
i i i i i yeah i think it i guess except the better
filters because i sent it to everybody just blew
it off okay it's really simple that
```

Figure 1.1: ASR output from a meeting excerpt.

```
we i uh i i you didn't get it i don't think i did i guess
these have got better filters because i sent it to everybody
you uh just blew it off okay it's really simple though
```

Figure 1.2: Human transcription of the meeting excerpt of Figure 1.1.

other tasks such as speaker diarization and speaker change detection. Finding the correct positions of the sentence boundaries can be seen as a binary classification task by deciding that for every word boundary, either a period has to be inserted (*sentence boundary*) or not (*non-sentence boundary*). We mentioned above that the errors made in the word recognition by the ASR make this task tougher on speech than it is on textual data. But the errors of the ASR are not the most challenging part for the sentence segmentation task. The most challenging part of sentence segmentation from speech comes from the nature of speech and its emission by humans.

Being produced by humans, speech is prone to variability created by the differences of gender, accent, education and emotional state of the speaker as well as by the environment (Shriberg, 2005; Cetin & Shriberg, 2006). The variability in speech makes it difficult for a machine to find patterns that indicate sentence boundaries. One could easily understand some of the issues met by a machine with speech by thinking of how often people begin a sentence and start a new one without having finished the previous one, how often people stammer, or how often they interrupt each other in the case of multi-speaker environment.

The frequency of the various disfluencies in speech depends on the context (number of speakers, formal/informal environment, etc.) in which the speech takes place. For example, disfluencies are expected to occur more frequently in a meeting, when several people are involved in a discussion, than in a broadcast news show, where there is most of the time only one main speaker. We designate the context and the particularities of a given speech as the *speech style*.

As mentioned above, sentence segmentation is a binary classification task; to perform a classification task, one needs to train a classifier on data that have been labeled. Usually, the amount of data used to train an efficient classifier for sentence segmentation is large; at the same time, labeling data is a very time consuming and labor intensive task that has to be done by humans.[1] Therefore, the labeling effort is usually reduced by using data that have already been labeled to train a classifier. This is typically what is done in language models training, where the language models are often trained on large amount of data that do not all come from the same corpus. The differences between speech styles make it inefficient to simply merge data that come from two different domains. In such a case, the classification model of one domain needs to be *adapted* to the second domain, in order to better match the characteristics (frequency of sentences, vocabulary, etc.) of the speech style of the second domain. The domain whose model is adapted is called the *out-of-domain* and the domain to which it is adapted is called the *in-domain*.

In the previous case, we made the assumption that very few data from the in-domain was available. However, adaptation can also be useful when a large amount of data has already been labeled. Depending on the similarity of the speech styles of the in-domain and out-of-domain and on their respective amount of data, using the out-of-domain classifier may lead to a reduction

[1]While the work load of labeling data is indeed a problem in general, it must be mentioned that some exceptions have recently appeared. For example, the idea of making the labeling task entertaining is a new field that researchers are exploring and has been applied to the labeling of images for the web. In this particular case, the labeling task is presented as an online peer-to-peer game and has been shown to be very successful both in terms of participation from players and in the quality of the labeling. More details on this can be found on and in von Ahn (2006).

Speaker 1:	`I I I I I.`
Speaker 2:	`Yeah.`
Speaker 2:	`I think it.`
Speaker 3:	`I guess except the better filters.`
	`Because I sent it to everybody.`
Speaker 3:	`Just blew it off. Okay.`
Speaker 1:	`It's really simple that.`

Figure 1.3: ASR output enriched with capitalization, sentence segmentation and speaker turn change.

of the classification error achieved by the in-domain classifier. Adaptation can thus be helpful to both *reduce the amount of labeled data needed* and *increase the accuracy* of a classifier.

In this work, first show the need for adapting the data among speech styles. We examine in particular adaptation among three speech styles: broadcast news, telephone conversations and meetings. Various adaptation methods to build mixed classifiers that perform sentence segmentation on speech transcriptions are used to adapt telephone conversations to meeting speech style. With the aim of understanding better the differences between the speech styles, we provide statistical measures on the data of each speech style and perform a cross-classification for broadcast news and meeting speech style using four combinations of lexical and prosodic features.

Structure of this work

In the next chapter, we give an overview of the speech recognition process; in particular, we explain how to extract the textual data on which the sentence segmentation is performed from the audio. We motivate the utility of sentence segmentation and discuss the concept of sentence in the context of speech. The last part of this chapter covers the prior work done in the field of automatic sentence segmentation.

Chapter 3 introduces the concept of adaptation and how it can be used to improve the accuracy of a classification task. We describe the most popular adaptation methods and report how they have been used in the speech and language processing domains.

In this entire work, we focus and use only one learning algorithm, the *Boosting* algorithm. Chapter 4 gives a detailed presentation of this algorithm and explains how we use it for the particular task of sentence segmentation.

Chapter 5 is devoted to the theoretical description of the adaptation techniques used in this work.

Chapter 6 is mostly dedicated to the practical description of the sentence segmentation problem. In particular, it describes the data sets and the metrics that are used in the experiments.

In Chapter 7, we present the results of the various experiments that were performed.

Finally, Chapter 8 summarizes the results discovered in this work and relates them with the future challenges of sentence segmentation and adaptation.

CHAPTER

2 Sentence Segmentation

In this chapter, we first formalize the sentence segmentation task as a binary classification problem. We then discuss the concept of sentence and the differences between textual data and speech transcriptions. The way speech transcriptions are extracted from the original audio wave of the speech is described in the last part of the chapter, preceding an overview of the sentence segmentation prior work.

2.1. Sentence Segmentation

While recognizing the words was a big deal at the beginning of speech recognition, current recognizers have become better and better in performing this basic task. Their increased accuracy allows us to look toward new challenges for the speech domain, such as parsing, automatic machine translation, information extraction, question answering, topic detection and summarization, which are all current and future promising research topics (Mrozinski *et al.*, 2005; Makhoul *et al.*, 2005; Shriberg *et al.*, 2000).

These tasks however, require more information than the sequence of words output by the ASR, and the output of the ASR thus has to be enriched. For example, in the automatic summarization task, one approach is to detect the sentences that contain the most information, to extract them from the original text and to combine them together as a summary (Mrozinski *et al.*, 2005). To extract sentences from the original text, the system has to know where the sentence boundaries are. Thus, when done on speech transcriptions, automatic summarization needs the sentences to be detected first by a sentence segmentation classifier.

Sentence segmentation is part of the enrichment process (also called meta-data extraction) that consists of adding information to the speech transcription of the ASR to guide further high-level textual processing tasks. Other tasks of the enrichment process are speaker diarization, speaker identification or speech activity detection, among others. Two excellent examples of rich transcription systems are the one built jointly by ICSI and SRI (Liu *et al.*, 2006) and the one developed by IBM (Soltau *et al.*, 2005).

2.1.1 Formalization as a Classification Problem

The role of sentence segmentation is to determine, given a sequence of words, where sentence boundaries should be inserted. Sentence boundaries can appear between each two words of a sequence of words. In other words, for each word boundary, a decision has to made whether a sentence boundary should be inserted or not. Sentence segmentation can thus be seen as a *binary classification task* where the instances to classify are the word boundaries and the class to which an instance is assigned is either the sentence boundary class C_s or the non-sentence boundary class C_n. We will explain further more in details how the classifier hypothesizes either one of the two classes.

As explained in the introduction of this chapter, most high-level speech processing tasks assume the presence of sentence boundaries. Moreover, sentence boundaries and punctuation in general are also useful for human readability. However, while people agree on the importance of finding the sentence boundaries, the concept of "sentence" is not always clear. In the next section, we discuss the issues related to the definition of what is a sentence.

2.1.2 Definition of a Sentence

Although the concept of "sentence" is the cornerstone of most languages, giving an accurate definition of it is difficult. Sentences have a structure that follows grammatical rules. For example, one of the most basic grammatical rules to form a correct sentence is "subject-verb-object". A simple illustration of this rule is shown in Figure 2.1. Grammatical rules can then be recursively extended; for example, the "object" part of the sentence can be composed of more than a noun phrase. Thus, one possibility of adding a prepositional phrase in the object of the sentence of Figure 2.1 is the sentence of Figure 2.2. Such a syntactically rule-based grammar description is easy to express to a computer and has been widely studied in the field of textual processing (Allen, 1995; Baker, 1979).

```
The cat [S] eats [V] the mouse [O].
```

Figure 2.1: Illustration of the grammatical rule "subject-verb-object".

```
The cat [S] eats [V] the mouse under the table [O].
```

Figure 2.2: Same as Figure 2.1, but the object of the verb has been extended.

While useful for text applications, the rule-based grammar approach can however not be used in the speech domain. The main reason for this is that people do often not form grammatically correct sentences when they speak. Shriberg (2005) describes the way people talk, examining in particular the various disfluencies such as filled pauses, repetitions, repairs and false starts, and the influence of emotion on the speech. Another element that tends to increase the variability of speech is the presence of multiple speakers. Indeed, when people talk to each other, they very rarely talk one after the other with no overlap at all. Cases in which people speak at the same time are frequent, as presented by Cetin & Shriberg (2006). Their analysis done on 26 meetings shows that on average, 12% of the overall foreground speech is overlapped by speech from one or more talkers. For instance, attendees often try to grab the floor by saying things like "`Yes, but`" or "`Okay, but`", which are designated as *floor grabbers*. *Floor holders* are the opponents to the floor grabbers and try to hold the floor by filling the blanks, which is usually manifested by a repetition of the same word or the emission of a noise such as "`um`". In a more peaceful way, one can also show when he agrees with the current speaker by uttering little words such as "`I see`", "`Uhhuh`", "`Right`", which are known as *backchannels*. Finally, people sometimes begin a sentence, realize that this is not the way they want to express whatever they want to express, and then begin the sentence again in another way, which are referred to as *disruptions*. Floor grabbers, backchannels and disruptions are called *dialog acts* and more details on them can be found in Zimmermann *et al.* (2006b).

The units generated by the segmentation into dialog acts can be, as shown with the above examples, syntactically incorrect and should thus not be considered as sentences. On the other hand, if we do not consider them as sentences, we will end up with longer sentences that do not make sense either, such as the one shown in Figure 2.3. In this example, the regular sentence "`I think we should decide option one instead of option two`" is interrupted by the floor grabbers "`okay but`" and the backchannels "`right`", "`okay`" emitted by other speakers. The correct punctuation according to the dialog act segmentation is shown in Figure 2.4.

```
I think we should decide option one right okay instead of option okay two but.
```

Figure 2.3: A sentence where neither backchannels nor floor grabbers are followed by a period.

```
I think we should decide option one. Right. Okay.
Instead of option Okay two. but.
```

Figure 2.4: Same sentence as in Figure 2.3 with backchannels and floor grabers followed by a period.

This example points out that even though a segmentation according to dialog acts would lead to syntactically incorrect sentences, it makes more sense to segment the sequence of words according to dialog acts than to the whole sentences, which is often not syntactically correct anyways. The segmented parts created according to dialog acts are called *sentence units* (SU). The five dialog acts that are considered and marked as sentence units are backchannels, disruptions, floor grabbers, statements and questions. In the rest of this work, we use the term "sentence" to designate the sentence units formed by the dialog acts.

The definition of the sentence through the five dialog act classes allows the sentence segmentation classifier to know what it is looking for. The next challenge is to find an appropriate learning algorithm to train the classifier and a set of features to represent the data, such that the sentence segmentation can be performed accurately. Section 2.3 gives an overview on how these two issues have been dealt with in the past, while the next section explains the practical process that leads from audio to a sequence of words.

2.2. Speech Recognition Overview

As formulated by Jurafsky & Martin, the purpose of speech recognition can be expressed as:

"What is the most likely sentence out of all sentences in the language $\mathcal{L}$ given some acoustic input O?"

Although this looks like a rather simple question, the machines that currently answer it the most accurately (ASRs) are a succession of non-trivial processes, which are described below.

2.2.1 Acoustic Analysis

The first step of the speech recognition process is to make the signal captured by the microphone, independent from the environment in which it was recorded (noise, reverberations, type of microphone, etc.). Ideally, the audio source ends up being independent from any non-lexical information and the lexical information can then be extracted.

Since the speech signal is a continuous and non stationary signal, it also needs to be decomposed into a digital signal. This is typically done by slicing the audio signal into overlapping *windows* of 20ms to 30ms every 10ms. Each of the windows is assumed to be stationary. An acoustic vector is derived from each of the windows, and all windows vectors are then concatenated to form a sequence of vectors. The final representation of an audio wave is thus a sequence of acoustic vectors that pictures the wave form at each 10ms interval. The acoustic input or observations O mentioned in the question at the beginning of this section are nothing but this sequence of vectors.

2.2.2 From Acoustic Vectors to a Sequence of Words: General Approach

According to the question at the beginning of this section, the goal is to identify the sequence of words $\hat{W}$ that is the most likely to occur according to a given observation vector O obtained from the acoustic analysis. This can be expressed as:

$$\hat{W} = \underset{W_j \in \mathcal{L}}{\operatorname{argmax}} P(W_j|O). \tag{2.1}$$

Since the number of observations O is small compared to the number of possible sentences, estimating Equation 2.1 is difficult. Bayes' rule can be used to rewrite Equation 2.1 as follows:

$$\hat{W} = \underset{W_j \in \mathcal{L}}{\operatorname{argmax}} \frac{P(O|W_j)P(W_j)}{P(O)}$$

Since $P(O)$ is independent of the hypothesis $\hat{W}$ (O is the same for every possible sentence), the above equation can be reduced to:

$$\hat{W} = \underset{W_j \in \mathcal{L}}{\operatorname{argmax}} P(O|W_j)P(W_j) \tag{2.2}$$

The unknown sequence of words W_j that we are trying to find can be modeled by a hidden Markov model (HMM). $P(W_j)$ and $P(O|W_j)$ depend on the parameters Θ of the HMM model, which makes Equation 2.2 become:

$$\hat{W} = \underset{W_j \in \mathcal{L}}{\operatorname{argmax}} P(O_j|W_j, \Theta)P(W_j|\Theta). \tag{2.3}$$

The previous equation makes the simplifying assumption that the set of parameters Θ is the same for all sentences W_t. This is a practically required assumption since it would be impossible to have different parameters, and thus different models, for each possible sentence. To match with this assumption, the models representing the sentences are decomposed into smaller lexical units. While in the very early speech recognizer systems the lexical unit was the word, most of the recent ASRs use phonemes as lexical units (Gadde *et al.*, 2002).

Up to now, we have examined the *recognition* part of the speech recognition task, which takes two input arguments: the acoustic observations O and the HMM model described by its parameter Θ. However, we have not explained where the probabilistic models that allow to compute the most probable sequence of words $\hat{W}$ come from. In fact, the probability models need to be *trained* before they can be used in the above equations. Equation 2.3 contains two probabilistic models $P(O|W_j, \Theta)$ and $P(W_t|\Theta)$ and can be decomposed according to them. As we will see, this split is appropriate since $P(O|W_j, \Theta)$ and $P(W_t|\Theta)$ each models one part of the problem.

The likelihood $P(O|W_j, \Theta)$ can be seen as the acoustic contribution and can be estimated with HMMs. During the training phase, the HMM parameters are optimized to maximize the likelihood of the set of training sentences given their associated HMM model. The training of the HMM parameters is beyond the scope of this work and the reader is referred to (Jurafsky & Martin, 2000; Boite *et al.*, 1999, among others) for more details on them.

The second part of Equation 2.3, $P(W_j|\Theta)$, is the probability of having a particular sequence of words. Since this probability has no dependence on the observed acoustic vector O, it can be estimated *a priori* on lexical only dependent data, typically using a Markov model. The independence of the sequence of words W_j with respect to the observed acoustic model O allows to split the parameters Θ into two subsets, the language dependent parameters Θ_L (called the *language model*) and the acoustic dependent parameters Θ_A (called the *acoustic model*).

The language model is often simplistically seen as a set of N-grams with associated probabilities that have been estimated on large corpora using only statistical computation (Markov model). The language model can nevertheless also contain syntactic and semantic features. In all cases, the language model is used to find the most probable sequence of words. In particular, given a sequence of words, it can be used to output a list of the most probable following words with a probability associated to each possibility.

The split of the parameters Θ into the two independent sets Θ_A and $\Theta_L$is a simplifying assumption, since in the reality a word is often pronounced differently depending on its position in a sentence. However, this simplifying assumption is particularly accommodating in the practice, since it allows to train the language model on textual data, and thus enables to increase the size of the training data. According to this split of the parameters Θ into Θ_L and Θ_A, we can rewrite Equation 2.3, that expresses the recognition task, as:

$$\hat{W} \approx \underset{W_j \in \mathcal{L}}{\operatorname{argmax}} P(O|W_j, \Theta_A) P(W_j|\Theta_L). \tag{2.4}$$

2.2.3 STT vs. Reference Conditions

Recall that in order to be able to train a classifier that detects sentence boundaries, one needs to have sequences of words annotated with the sentence boundaries. The annotation is done by humans who *listen* to the original audio recording of the speech and write down what they hear, annotating their transcription with punctuation. The transcription done by humans for a given audio wave is called the *reference* transcription. The reference transcription W_{REF} annotated by humans is assumed to contain the words that were indeed uttered by the speakers. In practice this is not exactly the case since humans can make mistakes; however, these mistakes can be partly removed by having different persons labeling the same audio source. Moreover, the human transcription of the speech is the transcription that is the nearest from the truth that one can obtain. Another type of transcription, the *speech-to-text* one (*STT*), is obtained from the sequence of words W_{ASR} output by the ASR.

Although the probability that the sequence of words W_{ASR} corresponds to the reference sequence of words W_{REF} has been optimized by the process described in the previous section, it is often the case that W_{ASR} and W_{REF} do not exactly match. As explained in more detail further in this section, W_{ASR} can in particular contain words that are not in W_{REF} (insertion), can miss words that are in W_{REF} (deletion), or replace words of W_{REF} by other words (substitution).

While the annotation has been done on the reference conditions, the STT conditions are the ones that correspond to the reality and under which the classifier should also be trained. The STT transcriptions also have to be annotated, but annotating data is a time-consuming and labor-intensive task, and one therefore wants to avoid to have humans annotate the STT conditions. Instead, an automatic alignment between W_{REF} and W_{ASR} can be done that allows the STT transcription to be automatically annotated with the tags from the reference conditions. The alignment process, which we explain in more detail below, is also used to complete the time information that is missing in the reference conditions.

Reference conditions

For the reference conditions, the speech is transcribed by humans into an enriched sequence of words that contains a tag for each word of the transcription. The tag can be one of the five dialog acts if the word marks the end of a dialog act, or a tag indicating that no dialog act ends after the word. No information about the timing is known at this point. This information is obtained through a procedure called *forced alignment*. Forced alignment takes as input the correct words along with the spectral feature vectors and produces the optimal sequence of phones. It is called *forced* because the correct sequence of words is given and the algorithm only has to find the best sequence of phones according to the pronunciation of the given words and the feature vectors. Once this has been done, the beginning and end times of each word can be added to the transcriptions.

STT conditions

For the STT conditions, the recognition process described in Subsection 2.2.2 is used to obtain the sequence of words: feature vectors are extracted from speech, phone likelihoods are estimated and turned into words. The problem is that the sequence of words issued from the automatic recognition process does not contain any punctuation information, which is needed in order to train and test classifiers for sentence segmentation. Asking humans to label it would turn out to do again what has already been done for the reference conditions. In order to spare this work and since the sequence of words in the reference and STT conditions is mostly similar despite the ASR errors, the correct tags of the STT words can be assigned by proceeding to an alignment between the correct sequence of words of the reference conditions W_{REF} and the guessed sequence of words of the STT conditions W_{ASR}. This task is however not trivial because the ASR is not perfect, deleting, inserting and substituting words during the word recognition phase. Taking these errors into account, the alignment algorithm uses dynamic programming and finds the alignment that minimizes the number of mismatches between the two sequences of words. This leads to a bijective function between each sequence of words. Each word in W_{ASR} is then assigned the same tag (one of the five dialog act tags or the non-dialog act tag) as its corresponding word in W_{REF}.

Problems caused by insertions, deletions and substitutions

The two above-described tasks of matching dialog acts tags to words of the STT transcriptions and matching beginning and end times to words of the reference transcriptions are made tougher because of the *insertions*, *deletions* and *substitutions* that ensue from the alignment process. For instance, it is often the case that the ASR misses a word (deletion), converts some background noise into a word (insertion) or detect the presence of a word without being able to determine the correct word (substitution). During the alignment with the reference sequence of words, the words that are missing in W_{ASR} are added, but there is no time information for them since the ASR did not notice them. On the contrary, the words that were inserted by the ASR have time information but since there is no corresponding dialog act tag in the reference transcription for them, their dialog act information is missing. In such cases, we associate a non-sentence boundary to the inserted word. In case of a substitution, both time and dialog act informations are present but the word is incorrect or missing (replaced by a `@reject@` tag).

These errors in the alignment process then result in a lack of information in the final STT and reference transcriptions, which have to be handled during the training phase. The long-term goal of the whole speech recognition process is to tend to a maximal automatization. So for consistency, the data on which the classifier is trained should correspond to the data that come out of this automated process, i.e. the ASR output. For this reason the sequence of words that is given to the classifier in both STT and reference conditions for the learning, does not contain the words deleted by the ASR, while it contains the words that the ASR falsely inserted. Once again, this can appear to be suboptimal since the sequence of words on which the classifier is trained is not the real one. Nevertheless, as the classifier is meant to be evaluated on the ASR output, it makes more sense to train it on the same kind of data that it will be evaluated on, even if these data do not exactly match with the original text. However, it should also be noticed that even with the accurate speech transcriptions of the reference conditions, the sentence segmentation on the reference transcriptions cannot be compared to the sentence segmentation task on textual data for speech fundamentally differs from text, as explained in Subsection 2.1.2.

2.3. Related Work on Automatic Sentence Segmentation

2.3.1 Acoustic vs. Linguistic Segmentation

A computationally efficient way of segmenting a sequence of words is to perform the segmentation at the audio level. In such a case, the ASR segments the wave form into acoustic sentences according to acoustic features, such as long silences or speaker changes. The problem with such segmentations is that they do in general not match with the *linguistic segmentations* done by humans (e.g. the ones described above with the dialog acts). For example, it is possible that the speaker makes a break in the middle of a sentence, in which case what is actually one sentence would be split into two sentences by the ASR; or that two acoustic sentences are merged because the speaker does not make any significant break between two sentences. Stolcke & Shriberg (1996) and Meteer & Iyer (1996) have shown the relevance of performing linguistic segmentation instead of acoustic segmentation. They used a combination of hidden Markov models and N-gram language models (HELM for *hidden event language model*) that outputs a probability of having a sentence boundary between any two words. The probability is then converted to an actual sentence boundary or a non-sentence boundary using a thresholding process, i.e. if the probability of seeing a sentence boundary is higher than a given threshold, a sentence boundary is inserted.

2.3.2 Classifiers

The HELM method was also used by Gotoh & Renals (2000) and by Shriberg *et al.* (2000) and extended with confusion networks (Hillard *et al.*, 2004; Zimmermann *et al.*, 2006a). Confusion networks were used in this case to take advantage of different word hypotheses from the ASR to reach a more accurate sentence segmentation. While all these works rely on a combination of language models and HMM, other classifiers have also been used. Zimmermann *et al.* (2006a) have compared various classification algorithms, such as boosting on weak learners, HELM, maximum entropy and decision trees. Their results showed that the best sentence segmentation performance was reached by a combination of the HELM classifier and a boosting algorithm on weak learners, originally designed for textual classification.

In a recent study, Roark *et al.* (2006) presented a new approach that did not use Markov models. Their main assertion was that Markov models are not able to model sufficiently long sequences to be efficient on the sentence segmentation task. In their approach, they used the maximum entropy model presented by Liu *et al.* (2006) to produce n-best lists of possible segmentations and extracted disambiguation features to choose the best one among them. Their new approach allowed them to use some features that had not been used in the above-presented methods, including statistical and syntactic ones.

2.3.3 Features

While these syntactic and statistical features have not been used before, at least two other main types of features have been widely studied for the problem of sentence segmentation: the lexical ones and the prosodic ones.

Lexical features are represented as word N-grams with sentence boundary. Word N-grams are naturally well shaped by the HELM method described in the previous paragraph, but they can also appear in other classifier types such as boosting. The main advantage of lexical features is that they are not limited to audio training data but can also be extracted from text corpora, and can therefore benefit from a larger amount of data than prosodic features.

Indeed, *prosodic features* are by definition related to oral characteristics such as rhythm, stress and intonation and hence require an audio representation of the data. Prosodic features have been used by Shriberg *et al.* (2000) and Liu *et al.* (2005), among others, and the results showed that prosodic features reduced the sentence segmentation error when added to the lexical features. In particular, pause duration has been shown to be one of the most important feature for the sentence segmentation task.

As mentioned in the previous paragraph, *syntactic features* have recently been used in a re-ranking approach and led to promising results (Roark *et al.*, 2006). In another attempt to find new features, Eisenstein & Davis (2005) studied the effect of gestural cues for sentence segmentation. They showed that gestures were indeed good indicators to detect the sentence boundaries, but that they did not improve the sentence segmentation performance when used together with lexical and prosodic features, since the information that they contained was already conveyed in the lexical and prosodic models used. However, this redundancy with the lexical and prosodic features makes gestural features a promising field for the cases where speech or prosody data are noisy.

2.3.4 Utility of Sentence Segmentation

The last two works mentioned herein may be the most important ones, since they showed the importance of performing sentence segmentation. Mrozinski *et al.* (2005) showed that automatic summarization performs better when done on data on which the punctuation is correct. They used different language styles corpora (broadcast news, conference proceedings and lectures), extracted the sentences that had the most significance and put them together to form a summary. They used random, automatic and human labeled sentence segmentation methods and showed that the more accurate the sentence segmentation, the most relevant the summarization. Makhoul *et al.* (2005) evaluated two tasks of the information extraction domain, entity and relation extraction, as a function of the punctuation. They showed that even if the most important factor for a efficient entity and relation extraction was the word error rate (WER) performed by the ASR, using automatic boundary detection instead of full correct punctuation, considerably reduced the score of both entity and relation extraction.

CHAPTER

3 Adaptation

This chapter first defines terms related to model adaptation and machine learning in general. We then describe the concept of model adaptation and justify the use of adaptation for classification tasks. A formalization of the model adaptation is then proposed, followed by the examination of two statistical frameworks (MLE and MAP), as well as how and in what fields they have been used for adaptation purposes.

3.1. Definitions

In the rest of this work, we will use some words that are specific to machine learning or to model adaptation. Since different terminologies are used in literature, we specify the meaning of some expressions that will be used throughout this work.

Classifier. A classifier is the result of the learning process. It is a predictive function $c = f(x)$ that, given an instance x and a set of classes C, associates one class $c \in C$ to the instance x, i.e. it *classifies* the instance.

Labeled or annotated data. Labeled/annotated data refers to a set of data in which each instance in the data set is manually assigned a corresponding class. The class of each instance is considered as the true one for that particular instance. Unlabeled/unannotated refers to the instances of the data that do not have a class assigned by a human labeler.

In-domain and out-of-domain. We consider a domain as a space that contains data which come from the same source. A domain can be a corpus of data, but as we will see from the MRDA adaptation experiment, it can also be a subpart of a corpus. With the task of adaptation, we distinguish two domains: the in-domain and the out-of-domain. The in-domain refers to the domain on which the classification task has to be applied. The out-of-domain refers to another domain which has no formal relationship with the in-domain, but can however be thought of as being helpful to classify data of the in-domain. Usually, the out-of-domain contains significantly more data than the in-domain and is used to create a general model, while the in-domain data are used to adapt the general model to a model that reduces the mismatch between the general model and the characteristics of the in-domain.

Training or building a classifier. Feeding a learning algorithm $\mathcal{L}$ with some training data D_t results in the creation of a classifier. We call this process the *training* phase. We also can say that the resulting classifier is *built* on D_t using $\mathcal{L}$.

Testing or evaluating a classifier. Once a classifier has been trained on some data D_t it can be *evaluated* on another set of data D_e, with $D_t \cap D_e = \emptyset$. In the evaluation phase, also called testing phase, the classifier attributes a class $c_e \in C$ to every instance $x \in D_e$. The class c_e that was attributed by the classifier to the instance x is then compared to the true class c_t of the instance x. If the two classes match, i.e. $c_e = c_t$, the classification of x is considered correct. In the opposite case ($c_e \neq c_t$), the classification of x is considered erroneous. Different metrics can then be used to evaluate the classification performance, based on the number and the type of instances erroneously and correctly classified.

In the rest of this work, a typical sentence in the description of the experiments will be: "The performance of the MRDA classifier is tested/evaluated on SWBD". This means that a classifier has been built on the training data of the MRDA corpus using the *Boosting* learning algorithm and its performance is evaluated on the test set of the SWBD corpus.

3.2. Definition and Usage of Adaptation

As explained in the previous chapter, our approach to the sentence segmentation is to consider it as a binary classification problem (i.e. two classes), in which *every word boundary* has to be classified as a sentence boundary or as a non-sentence boundary. In the statistical learning process, one first needs to choose a learning algorithm. In this work, we mainly focus on a boosting algorithm which we describe in more detail in Chapter 4, as it was shown to yield top-tier performances for sentence segmentation (Zimmermann *et al.*, 2006a) as well as text classification tasks (Gupta *et al.*, 2006). Once a learning algorithm has been selected, the next step is to feed this algorithm with a set of training data. The data set is split into *instances* (also called *examples*) which are represented by a set of features and a class. For the sentence segmentation task, the features are mainly word N-grams and prosodic characteristics, as presented in Section 2.3. In order to be able to learn how to classify the instances according to their features, the algorithm needs to be provided with the true class of the training instances, i.e. with instances that have been labeled.

Labeling the data, i.e. attributing the correct class to each instance of the data set, has to be done by humans, since they are the only ones able to define what a sentence is according to the complex definition given in Subsection 2.1.2. This is a time-consuming task which has to be done meticulously and following a number of well-defined rules to make the data as consistent as possible.[1]

Adaptation is a general concept which can be employed to reduce the human labeling effort by reusing existing labeled data, even when these existing data do not exactly match with the domain of the data that need to be classified. Indeed, large amounts of data have already been labeled in the past years through various projects in the spoken language processing field. Some of these sets of data, called *corpora*, are publicly available. Adaptation is the process of using these already labeled data (*out-of-domain*) to help build or improve a classifier for the new domain (*in-domain*). We distinguish two adaptation methods: *unsupervised adaptation* and *supervised*

[1] An example of these rules are the labeling guidelines for the ICSI meeting project (Dhillon *et al.*, 2004).

adaptation. Unsupervised adaptation refers to the case where no labeling is provided for the in-domain data. The only labeled data are thus the data of the out-of-domain.

The data of the out-of-domain are however not specific to the in-domain and can even be different from the in-domain data. Thus, classifying the in-domain data with a classifier built on out-of-domain data could result in a low accuracy classification. To make the classifier more closely related to the in-domain, one may therefore want to include some knowledge of the in-domain data into the classification model construction. If so, some in-domain data thus have to be manually labeled. In this case, the classifier is provided with some knowledge of the in-domain and the adaptation is then known as supervised in this case.

3.2.1 The Choice of the Out-of-Domain and the In-Domain

Adaptation requires the data of the in-domain and the out-of-domain to share some similarities. It would, for example, make no sense to perform language model adaptation from one corpus in English to another one in Arabic. The exact amount of similarities is however difficult to evaluate, although some statistical measures from the information theory (mutual information, KL-divergence, etc.) can be used to this end. Percepting the relation between the in-domain and the out-of-domain data can also lead to selective sampling techniques, where the samples containing useful knowledge are uppermost selected.

Besides quantitative assumptions, one could also make some qualitative assumptions on what has to be the same for both the out-of-domain and the in-domain data. For example, the number of classes in both domains should be related, i.e. either be the same or at least have a common identical subset. Another requirement would be that the data had been labeled following the same guidelines, or at least if the labelings are different, that they could be made consistent together. The latter point is especially important in the context of sentence segmentation, because as we have seen in Subsection 2.1.2, defining what is a sentence boundary (and what is not), is not always straight forward.

The relation between the out-of-domain and the in-domain can change depending on the task for which adaptation is used. The out-of-domain can be a domain which has the same major characteristics as the in-domain, but for which some characteristics differ from the in-domain. The adaptation of telephone conversations on meetings that is studied in this work is such a case, since telephone conversations and meetings share the similarity of both being conversational speech but differ in the number of speakers and the possibility of seeing one's interlocutor, among others. Another way of considering adaptation is to take the out-of-domain as a general classification model and to adapt it to a more specific in-domain. This is the approach used when creating speaker-dependent language models: a general language model is built based on the words that a large population use and is then adapted to a more specific domain, i.e. the speaker-related domain (Souvignier & Kellner, 1998, among others). The previous example reveals that model adaptation is not limited to the sentence segmentation task. As we will see in Subsection 3.3.2, adaptation can be used in several problems across various domains.

3.2.2 Formalization

This section has been inspired by the work of Daumé & Marcu (2006).

In the classification problem, it is typically assumed that a set of data $D = (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ is available where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ is the finite set of labels. The data set D is usually split into three subsets: the training set, the test set and the development (or held-out) set. It is assumed that every sample (x_n, y_n) is drawn from an unknown fixed distribution p. The learning task consists of finding a predictive function $f : \mathcal{X} \to \mathcal{Y}$ from the training set, so that f minimizes the number of errors when applied on the test set, where errors are the cases in which the label $y_{e,n}$ guessed by f for an instance x_n is different from the true label y_n. The development set can be used to optimize parameters.

In the case of adaptation, we assume two sets of data $D^{(o)}$ and $D^{(i)}$ drawn from two different fixed unknown distributions, $p^{(o)}$ and $p^{(i)}$ respectively. $D^{(o)}$ stands for the out-of-domain data while $D^{(i)}$ stands for the in-domain data. The learning problem is to find a function that has a high predictive accuracy, i.e. that minimizes the number of errors on the in-domain test set, $T^{(i)}$. From a mathematical point of view, the problem could be inverted and the goal could be defined as finding a predictive function on the test set of the out-of-domain data. This is however not the case in practice, since the amount of labeled data available in the in-domain is much smaller than the amount of labeled data available in the out-of-domain: $|D^{(o)}| = N^{(o)} \ll N^{(i)} = |D^{(i)}|$.

In the introduction to this chapter, we mentioned that the in-domain and the out-of-domain should share similarities to enable adaptation. These similarities can now be formally expressed as the similarity between the two distributions $p^{(o)}$ and $p^{(i)}$. If the two distributions are the same, the two data sets can be merged into one single set and no adaptation is needed. If on the contrary the two distributions of the two domains are independent, then the knowledge of one domain brings nothing to the other one and adaptation is thus useless. When proceeding to adaptation, we therefore assume that $p^{(o)}$ and $p^{(i)}$ are neither independent nor identical.

In the case of sentence segmentation from speech, we typically adapt a classification model with one speech style to a domain with a different speech style. For example, as we will see further in this work, one sentence segmentation classifier can be built on telephone conversations and adapted to multi-party meetings. In this case, differences between the speech styles could be a different number of speakers and the possibility or not to see one's interlocutor, among others. These differences make the distributions of the sentence boundaries in the two domains not identical. However, since there also are a number of similarities among these two speech styles, such as the language (English) and the type of speech (conversational), the distribution of the sentence boundaries is not independent.

Being a general concept, adaptation requires some choices to be made to become applicable to a given problem. In particular, one has to choose:

- a set of relevant *features* to represent the data,
- a *learning algorithm*,
- a way to *adapt* the out-of-domain classifier to the in-domain.

Features. The set of features describing the samples to classify must be relevant in both the out and the in-domain. For example, in the case of the sentence segmentation task, if the pause duration between two words is the most decisive feature in the in-domain data whereas for some reason it is of no importance in the out-of-domain data, it is likely that a classifier built on the out-of-domain data will lead to an inaccurate classification on the in-domain data. Results using different sets of features are shown and discussed in Section 7.5.

Learning algorithm. The learning algorithm chosen should be appropriate to the task. For example, while support vector machines (SVM) yield top results in most of the learning tasks, (Zimmermann *et al.*, 2006a) have shown that the BoosTexter tool (Schapire & Singer, 2000), which implements a boosting learning algorithm, is particularly efficient for text categorization and sentence segmentation problems. Another work showed the efficiency of *Boosting* for adaptation tasks (Tur, 2005). We therefore use the BoosTexter tool in our experiments. Chapter 4 describes in more detail the learning algorithm implemented by BoosTexter.

Adapatation methods. The last choice one has to make is the way of combining the classifiers of the in-domain and the out-of-domain. The combination of both classifiers should exploit the maximal in-domain relevant knowledge from the two domains. We discuss different adaptation methods in Chapter 5.

The three presented choices have a big influence on the quality and the accuracy of the classification that will be obtained by our system. As mentioned for the choice of the learning algorithm, looking at what have previously been done is useful. In the next section, we review the work that has been done on this particular aspect of combining one out-of-domain model with one in-domain model to perform classification on the in-domain data.

3.3. Related Work on Model Adaptation

The most reported models of adaptation in the literature use either the *maximum a posteriori* (MAP) framework or the *maximum likelihood estimation* (MLE). In order to better understand how these two statistical models are used for the adaptation, we shortly recall the basic principles of both of them.

3.3.1 Brief Presentation of MLE and MAP

In both cases a set of data $X = \{x_1, \ldots, x_n\}$ is given and is assumed to be drawn from an unknown distribution $p(x)$. Our goal is to find the parameters θ of the distribution that match the best with the data X.

The *likelihood* for a particular sample $x \in X$ with respect to θ is defined as the joint probability function $p(x|\theta)$. MLE assumes that the parameters θ of the distribution are fixed but unknown. For example, we can assume that the data is distributed according to a Gaussian (or a mixture of Gaussians) for which we want to estimate the parameters $\theta = (\mu, \Sigma)$ that will make the Gaussian distribution the nearest from the data contained in X. Assuming that the data in X are independent, the likelihood turns out to be:

$$p(x|\theta) = \prod_{j=1}^{n} p(x_j|\theta)$$

In an intuitive way, MLE defines the estimator of θ as being the one that yields the largest likelihood $p(x|\theta)$:

$$\theta_{MLE} = \operatorname*{argmax}_{\theta} p(x|\theta) \tag{3.1}$$

θ_{MLE} can be found by deriving the log likelihood and setting it to zero.

The MAP model is part of the Bayesian estimators family, which have a completely different approach. While MLE assumed that the parameters θ were unknown but fixed, Bayesian estimation assumes that the parameters θ are also random variables with a prior distribution $p(\theta)$. Therefore, the posterior distribution that we want to maximize is now:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \approx p(x|\theta)p(\theta)$$

The parameter θ_{MAP} estimated by MAP can be directly expressed from this equation:

$$\theta_{MAP} = \operatorname*{argmax}_{\theta} p(\theta|x) = \operatorname*{argmax}_{\theta} p(x|\theta)p(\theta) \tag{3.2}$$

A comparison of equations 3.1 and 3.2 makes it clear that MLE is only a special case of MAP, where the prior distribution $p(\theta)$ of the parameter θ is uniform. The differences in the two approaches are especially important for the problem of adaptation and the cases where only a limited amount of data is available. Indeed, MLE requires an important amount of training data to achieve accurate results, since it makes no assumption on the parameters θ. On the contrary, the prior distribution $p(\theta)$ used in the case of MAP can be seen as a prior knowledge of the model (before any data has been observed) and can be useful in the case where only a little data is available.

This case is exactly the one of the adaptation problem, in which very little data of the target domain (in-domain) is available. The prior distribution $p(\theta)$ of the parameter θ can be estimated on the out-of-domain data. Denoting the data sets as $D^{(i)}$ and $D^{(o)}$ for the in-domain and the out-of-domain and their parameters as θ_i and θ_o respectively, Equation 3.2 becomes:

$$\hat{\theta}_i = \underset{\theta_i}{\operatorname{argmax}}\, p\left(D^{(i)}|\theta_i\right) \overbrace{p\left(\theta_i | \underbrace{\underset{\theta_o}{\operatorname{argmax}}\, p(\theta_o) p\left(D^{(o)}|\theta_o\right)}_{\text{MLE estimation on the out-of-domain}}\right)}^{\text{Prior distribution of } \theta_i} \tag{3.3}$$

As Huang *et al.* showed (2001), the weight of the prior knowledge can be formulated as a function of the amount of in-domain data available. Thus, as the amount of in-domain data increases, the MAP estimate tends to the ML estimate, since the in-domain data get a higher weight than the prior knowledge. Although the MAP approach seems very attractive, one should keep in mind that the estimation of $p(\theta)$ is a critical operation, especially because $p(\theta)$ may be a non trivial distribution and because the quality of the guess of $p(\theta)$ depends on the similarity between the out-of-domain and the in-domain, which is usually difficult to predict.

Another more subtle limitation of the MAP approach is that only the distributions of the parameters that appear in the in-domain data can be modified. This could thus lead to a very slow adaptation and require some extra means to have the poorly adapted parameters altered. So in some cases, it might be better to use the MLE approach instead of the MAP one. The adaptation of the parameter θ can be done using linear regression transformation functions (MLLR). An example of such an adaptation is shown in Huang *et al.* (2001, p. 450) for speaker adaptation using HMM with Gaussian probability density functions. Note that the MAP approach and the MLLR can be combined to take advantage of both methods. The reader is referred to Huang *et al.* (2001) for more details on MLLR and the inclusion of the MAP approach into it.

3.3.2 Prior Work

As far as we know, model adaptation has never been applied to the problem of sentence segmentation. Both supervised and unsupervised adaptation have however been shown to be useful in other speech and language related tasks, such as language models (LM) and probabilistic contextual free grammars (PCFGs) (Bacchiani & Roark, 2003; Roark & Bacchiani, 2003). The approach presented in these two articles is based on the above-presented MAP approach. For the LM adaptation, Bacchiani & Roark used the Dirichlet density to estimate the prior discrete distribution across words. By only changing the parameter of the prior Dirichlet distribution, they achieved two techniques, count merging and model interpolation. These two techniques were evaluated at 4 stages of the recognition process and both count merging and model interpolation outperformed the baseline in all cases. The MAP approach presented in this work has later been compared with a perceptron algorithm which re-weights paths in a lattice of words in (Bacchiani *et al.*, 2004). The experimental results of this study showed that the MAP approach outperformed in general the perceptron algorithm, but that the latter had the same performance as the MAP under certain conditions.

X. Fang & Sheng (2003) identified two main problems for the LM adaptation, which are the poor quality of the out-of-domain data and the mismatch between the N-gram distributions in the out-of-domain and the in-domain. To solve these two problems, X. Fang & Sheng used filtering techniques together with a linear interpolation between the out-of-domain and the in-domain LMs. The results showed a reduction of the perplexity.

The count merging and model interpolation techniques mentioned above for LM adaptation have also been used in the case of PCFGs (Roark & Bacchiani, 2003). The prior distribution was still estimated with the Dirichlet distribution, but the goal was to evaluate the left-hand side of each rule of the PCFG instead of the discrete distribution across words. The results showed that adaptation can achieve big improvements for statistical parsing.

Chelba & Acero (2004) investigated the MAP approach together with a maximum entropy classifier. The approach used is to train a maximum entropy model on the out-of-domain data and to use the optimal weights of this model as the mean weights for the Gaussian priors of the in-domain model. The task described in this work was to recover the correct capitalization of uniformly cased text. The adaptation was performed from the Wall Street Journal corpus to broadcast news data and showed a reduction error of 20% over the non-adapted version.

Another domain of speech processing where adaptation has been used is spoken language understanding systems. Tur (2005) used supervised adaptation for intent classification from one existing application to one similar new target application. He showed that using adaptation reduces the amount of labeled data required to achieve a given classification accuracy. In the same effort to reduce the amount of labeled data, he also showed that the same error rate can be achieved with a smaller amount of labeled data by selecting the examples that need to be labeled before to give them to the human labelers. This technique is called active learning and can be used together with supervised adaptation.

While all the above presented methods follow the most common approach of considering two different distributions for the out-of-domain data and the in-domain data as explained in Subsection 3.2.2, an interesting approach using *three* distributions has been presented by Daumé & Marcu (2006). The basic idea was to create one general distribution common to the in-domain and the out-of-domain models. This common distribution is then used in conjunction with the out-of-domain and the in-domain data to build truly domain specific distributions for the out-of-domain and the in-domain, respectively. Daumé & Marcu developed a new framework using the maximum entropy models and showed that the developed framework allows to reduce the classification error on language processing tasks significantly.

Another common approach is to build a classification model on the out-of-domain data and then use the prediction of this model as a feature for the in-domain. This approach has the advantage of being classifier independent, since only the prediction of the out-of-domain is used. Florian *et al.* (2004) used this approach for the entity detection and tracking problems. The results presented showed top-tier performance. Although Florian *et al.* did not explicitly measure the relative improvement of including predictions of out-of-domain classifiers to the in-domain data as feature, this tends to show the efficiency of the latter method.

CHAPTER 4

Boosting Algorithm

This chapter is dedicated to Boosting, the learning algorithm that we use throughout this work. We begin by retracing the history of Boosting and the mystery that surrounded it when it was first presented. We then go through the key ideas of the algorithm and explain how we use it for the sentence segmentation task. In the last part of the chapter, we examine in more detail the weak learners and the loss function used by Boosting, two concepts needed to transform Boosting in a convenient adaptive algorithm.

4.1. Key Concepts and History of *Boosting*

While the adaptation scheme presented in the previous chapter is a general concept, it requires in practice the usage of a classifier. In this work, we chose to use AdaBoost, an algorithm that uses boosting. Boosting is general concept in machine learning that consists of improving the final result of a learning algorithm by iteratively training a classifier on a weighted training set of data. At the end of the learning process, a weighted linear combination of the classifiers built at each iteration forms the final classifier.[1] *Boosting* is part of a larger theory called *ensemble learning*. While ordinary machine learning algorithms, such as decision trees or perceptrons, work by searching the best hypothesis for a set of data, ensemble learning algorithms have a different approach. Rather than finding a unique hypothesis, they construct a *committee* of simple hypotheses and combine them to obtain a final hypothesis.

When AdaBoost was first presented (Freund & Schapire, 1995), it immediately drew the attention in the machine learning community, since it was one of the most accurate classification algorithm available at that time. Researchers associated it to bagging at the beginning and thought that the major strength of *Boosting* had to do with variance reduction. But experiences showed that *Boosting* also reduces bias and thus differs from bagging (Friedman *et al.*, 1998). It has since then led to more research, both on the practical side and on the theoretical side and is still a widely and lively discussed topic in the machine learning field.

For example, one fascinating characteristic of *Boosting* is that it seems to be resistant to overfit-

[1] We used the particular AdaBoost.MH version of the algorithm, presented below. This is the algorithm which we will refer to as *Boosting* in the rest of this work and implicitly when mentioning boosting in general.

ting. This phenomenon had no theoretical explanation until 1998 when Schapire *et al.* found an upper bound error for *Boosting* and argued that the non-overfitting effect was explained by the fact that *Boosting* produced a high margin distribution. This demonstration was however voided by Breiman (1996) who presented an algorithm that had a higher margin distribution than *Boosting* and yet was less accurate. Recently Schapire *et al.* showed that their proof still holds if the classifier complexity is not increased (Reyzin & Schapire, 2006).

For the case of sentence segmentation, *Boosting* has been proven to outperform all other techniques to which it was compared, including decision trees, HELM and Maximum Entropy classifiers (Zimmermann *et al.*, 2006a). Being an efficient learning algorithm in general, *Boosting* is moreover appropriate for the sentence segmentation task since it handles word N-grams *and* at the same time accepts continuous features as well, which is needed for most of the prosodic features have real number values.

4.1.1 The Algorithm

In this study, the experiments were run with the BoosTexter tool developed by Schapire & Singer. BoosTexter implements different versions of the AdaBoost algorithm for the task of text categorization. The version used in this work has a discriminative power because it not only checks for the presence of a feature but also for the absence of it. As explained above, the idea of boosting is to run several times one simple classifier on a weighted training set of data, changing the weights at each iteration. More formally, at each iteration t, a new simple classifier h_t is built and all simple classifiers $h_t, t \in [1, \ldots, T]$ are combined at the end of the T iterations of the learning process into one final classifier $f(x, l)$:

$$f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l) \tag{4.1}$$

with α_t the weight of the weak learner h_t.

In the case of BoosTexter, the simple classifiers used are *weak learners*. In the case of *Boosting*, a weak learner has the same form as a one-level decision tree. The pseudo-code of the AdaBoost.MH algorithm is shown in Algorithm 1. The next paragraph summarizes the functioning of *Boosting*, but the reader is encouraged to examine Schapire (2001) and related papers for more details on this algorithm.

Each training example i is assigned L weights $D(i,l)$, i.e. one weight for each possible class ("s" and "n" in the case of sentence segmentation). At the beginning of the learning process, all weights are equal. At each iteration, a weak learner finds a weak hypothesis $h_t(x,l) : \mathcal{X} \to \Re$ given the distribution of the weights $D(i,l)$. The sign of $h_t(x,l)$ is interpreted as a prediction of whether the label l belongs to the set of labels assigned to the sample x ($h_t(x,l) > 0$) or not ($h_t(x,l) \leq 0$). Once this evaluation has been done for each label l, the following function $Y[l]$ can be described:

$$Y[l] = \begin{cases} +1, & \text{if } l \in Y \\ -1, & \text{if } l \notin Y \end{cases}$$

for $Y \subseteq \mathcal{Y}$ the set of labels attributed to x and $l \in \mathcal{Y}$. In the binary classification task, the membership sign $\in$ can be replaced by the equal sign since only one label can be attributed to each example. The weights are then updated such that the next iteration $t+1$ will force the weak learner to focus on the examples that were wrongly classified at the last iteration t. The final hypothesis is a weighted linear combination of the hypotheses of all iterations, with higher weights attributed to the weak hypotheses with a lower classification error rate.

Algorithm 1 AdaBoost.MH

Given the training data: $(x_1, Y_1), ..., (x_m, Y_m)$ where $x_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y}$
Initialize the distribution $D_1(i,l) = 1/mk, i = 1..m, k = 1..|\mathcal{Y}|$
for each iteration $t = 1..T$ **do**
Train a base learner h_t using distribution D_t
Update

$$D_{t+1}(i,l) = \frac{D_t(i,l)e^{(-\alpha Y_i[l]h_t(x_i,l))}}{Z_t} \tag{4.2}$$

where Z_t is a normalization factor and α_t is the weight of the base learner
end for
The final classifier is the weighted sum of the $h_t(x,l)$:

$$f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l)$$

4.2. From Continuous Predictions to Classification

The final output of the boosting algorithm described in Algorithm 1 yields a real number value $w = f(x,l)$ for each example x and label l. These $w = f(x,l)$ are the weights of the label l attributed to a sample x. To achieve the final goal of classifying the example x as a sentence boundary C_s or as a non-sentence boundary C_n, the weights $f(x,l)$ need to be interpreted and mapped to a class. The conversion from the weights output by the *Boosting* to the attribution of a class is done in two steps. First, each weight is turned into probability $p(l|x)$ according to the following equation:

$$p(l|x) = \frac{1}{1+e^{-2f(x,l)T}} \tag{4.3}$$

where T is the number of iterations used to train the classifier and $f(x,l)$ the final classifier output by the learning algorithm. These probabilities can then be converted into a binary classification by using a threshold, which we refer to as the acceptance threshold t_{acc}.

$$H(x) = \begin{cases} C_s, & \text{if } p(C_s|x) > t_{acc} \\ C_n, & \text{if } p(C_s|x) \leq t_{acc} \end{cases} \tag{4.4}$$

$H(x)$ designates the class of the example x. The threshold t_{acc} is a parameter and can be optimized on the development set. The effects of varying t_{acc} are discussed in more detail in Section 6.4.

4.3. More on Weak Learners

In the above-presented description of *Boosting*, the presence of a weak (or base) learner h_t was assumed. The role of the weak learner was to provide a weak hypothesis at each iteration. The weak hypotheses returned by the weak learner throughout the T iterations all have the form of a one-level decision tree. Even though the weak hypothesis $h_t(x_i,l)$ maps each sample x to a real value in $\Re$, it is easier to understand it in the simplest case where each $h_t(x_i,l)$ is binary, i.e. restricted to $\{-1,+1\}$. The base learner minimizes the probability that the wrong label will be attributed to a sample x for the current weights distribution D_t. This probability can be expressed by the following error function:

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq Y_i[l]] \tag{4.5}$$

For instance, in the sentence segmentation task, one wants to determine if a word is followed by a sentence boundary or not. For each example x, we also give to the algorithm the previous word. The weak learner then chooses the word w_p among all the "preceding words" set of words and create a basic rule saying "*if the word in instance x is preceded by the word w_p than give it value r_1, otherwise give it value r_2*". The choice of the word w_p takes into account the overall weighting error made by choosing this word. In the usual case where more than one feature is available, the best hypothesis obtained out of all features is returned.

Once the algorithm has been provided with a base classifier h_t, the next step is to determine the weight $\alpha_t \in \Re$ that will be attributed to h_t in the linear combination that forms the final classifier. In the case of a binary hypothesis $h_t(x_i, l)$, this is done as follows:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Thus, if the error ϵ_t inferred by the base classifier h_t is big, the weight α_t assigned to h_t is small. A small α_t results in a higher exponent for e in the weights update (Equation 4.2). If the hypothesis done by h_t was correct for the example x_i, y_i and $h_t(x_i)$ have the same sign and the exponent is thus negative, leading to a reduction of the weights $D(i, l)$ of the example x_i. On the contrary, if the hypothesis done by h_t was wrong, the signs of $Y_i[l]$ and $h_t(x_i)$ in Equation 4.5 are opposed, leading to a positive exponent and thus to an increase of the weight $D(i, l)$ of sample x_i. The sample x_i will therefore have a bigger importance in the next iteration when a new base classifier will be chosen.

4.4. Changing the Loss Function

Friedman *et al.* explained the phenomenon of *Boosting* in statistical terms (Friedman *et al.*, 1998). In particular, they showed that the *additive model* (also called *weighted committee* or *ensemble*)

$$f(x, l) = \sum_{t=1}^{T} \alpha_t h_t(x, l)$$

produced by *Boosting* is nothing but a Newton method for optimizing a particular exponential loss function. In a much shorter and more bare work, Fleuret showed that *Boosting* can be seen as a gradient descent, considering that at each iteration, the weak learner selected is the one that decreases the most the training error. The *Boosting* algorithm presented in Algorithm 1 minimizes the exponential loss function:

$$\sum_i \sum_l e^{-Y_i[l] f(x_i, l)} \tag{4.6}$$

where, $Y[l] = 1$ if $l \in Y$ and -1 otherwise. The two works that were just mentioned emphasize the fact that the loss function of Equation 4.6 is an abstract concept and thus can be substituted by another loss function. In particular, a logistic loss function can be used that has the following shape:

$$\sum_i \sum_l ln(1 + e^{-Y_i[l] f(x_i, l)}) \tag{4.7}$$

with the same symbols as for Equation 4.6. This will reveal to be particularly convenient for the *Boosting* adaptation method that we will present in Section 5.4.

CHAPTER 5

Adaptation Methods

In this work, we use the existing labeled data or models to improve the classification performance in a new domain. The combination of the two sets of data can be done at various levels, such as the data level, the classifier level or the score level. This chapter describes and discuss four adaptation methods that will be experimentally compared in Chapter 7.

5.1. Data Concatenation

Data concatenation is the simplest way of combining two sets of data: the classification model is trained on the concatenation of the out-of-domain data $D^{(o)}$ and the in-domain data $D^{(i)}$. By concatenation, we mean simply joining the two sets in one set that is the union $D^{(u)} = D^{(o)} \cup D^{(i)}$ of the two original sets. The classification model is then trained on the $D^{(u)}$ data set. An alternative to data concatenation is the tagged data concatenation, where an extra feature that contains the source corpus of the example is added. This method has been used in particular in Tur (2005).

Although data concatenation is a very basic way of performing adaptation between two sets of data, there are at least two reasons for using it. First, it can be used as an easy baseline result: whatever the adaptation method, if it performs worse than concatenation, it is not worth use it. Second, the result of a classifier trained on the concatenated data is an early indicator concerning the compatibility of the in-domain and the out-of-domain. Indeed, since the out-of-domain data are not weighted when added to the in-domain data, if they do not match with the in-domain data, they will hurt the classification performance.

5.2. Interpolation using Logistic Regression

Interpolation using logistic regression is a *score-level* adaptation method. The basic idea is to build separately the two classifiers, i.e. the out-of-domain classifier on the out-of-domain data $D^{(o)}$ and the in-domain classifier on the in-domain data $D^{(i)}$. Once trained, each classifier is used to evaluate a score for each example x_j in the form of a posterior probability $P(C_s|\, x_j)$ for x_j to be associated with a sentence boundary. The posterior probabilities $P^{(o)}(C_n|\, x_j)$ and $P^{(i)}(C_n|\, x_j)$ of the out-of-domain and in-domain respectively are then combined to form the final prediction $P_f(C_n|\, x_j)$ of x_j. The combination of $P^{(o)}(C_n|\, x_j)$ and $P^{(i)}(C_n|\, x_j)$ can be done in various ways. The most intuitive one may be the linear combination

$$P_f(C_n|\, x_j) = \alpha P^{(i)}(C_n|\, x_j) + (1-\alpha)P^{(o)}(C_n|\, x_j),$$

where α is a parameter that balances the importance of the in-domain and the out-of-domain, and can be optimized on the development set.

The method that we present here follows the idea of the previous paragraph, except that the final combination between the two predictions $P^{(o)}(C_n|\, x_j)$ and $P^{(i)}(C_n|\, x_j)$ is done using *logistic regression*. The expression of P_f is shown in Equation 5.1. The goal of logistic regression is to find a discrete outcome from a set of variables that might be discrete or continuous, called *predictor variables*. No assumption is made about these variables (e.g. they do not need to be normally distributed or have a linear dependency to some other variables). In our case, the outcome is dichotomous and indicates the presence or the absence of a sentence boundary. The predictor variables are the probabilities output by in-domain and out-of-domain classifier that an event is a sentence boundary. The relationship between the predictor variables and the output is described by Equation 5.1. We describe the framework of logistic regression in more detail in Appendix A.

Concretely, each sample of the held-out set is evaluated by the classifier $C^{(i)}$ trained on the in-domain data and the classifier $C^{(o)}$ trained on the out-of-domain data. These evaluations yield two sets of probabilities $P^{(i)}(C_n|x)$ and $P^{(o)}(C_n|x)$ that the event associated with the sample x is a sentence boundary according to the classifier $C^{(i)}$ (resp. $C^{(o)}$). These probabilities are used to optimize b_1, b_2, b_3 by minimizing the negative log likelihood of Equation 5.1, using the Newton-Raphson method which is shortly presented in Appendix A. The two same classifiers $C^{(i)}$ and $C^{(o)}$ are then used to estimate two probabilities for each instance of the test set. The final decision is made from the combination of these two probabilities using the logistic function with the weights optimized previously on the development set:

$$P_f(C_n|\, x) = \frac{1}{1 + e^{(-b_1 - b_2 P^{(i)}(C_n|\, x) - b_3 P^{(o)}(C_n|\, x))}} \tag{5.1}$$

where b_1, b_2, b_3 are parameters that were previously optimized on the held-out set.

In the rest of this work, we refer to this method as the "logistic regression" method.

5.3. Using Out-of-domain Model Prediction as an Extra Feature

$C^{(o)}$ is run first on the *training* set of the in-domain data; the result of this run is a posterior probability $P^{(o)}(C_n|x_j)$ of having a sentence boundary for each instance x_j of the in-domain data set $D^{(i)}$ according to the out-of-domain classifier. This probability is added as an extra feature to the data set $D^{(i)}$, so that the new feature set $\overline{F}$ becomes:

$$\overline{F} = F \cup P^{(o)}(C_n|x).$$

The final prediction is output by the classification model $C^{(i)}$ then trained on the training set of $D^{(i)}$ extended with the new feature. Note that the prediction of the out-of-domain model should not only be added to the training data set of the in-domain, but also to the test set so that the two feature set of the training data and the test data are the same.

In the rest of this work, this adaptation method is designated as the “feature” method.

5.4. *Boosting* Adaptation

Boosting adaptation has first been suggested in the case of a spoken language understanding system (Schapire *et al.*, 2005). Schapire *et al.*'s goal was to be able to provide some human-crafted *prior knowledge* to the *Boosting* model. Later on (and still as part of a spoken language system), this method was used by Tur (2005) to adapt intent classification from one spoken dialog application to another similar application. The basic idea of *Boosting* adaptation is that instead of directly building a classification model M from some data d, one first builds a model M_P based on some existing prior knowledge. By modifying the *Boosting* learning algorithm, or more precisely by modifying the loss function of the algorithm, a new model M_D is built starting from the M_P model and *adapting* it to the data d. The next paragraph shows the details of *Boosting* adaptation.

First, as explained in Section 4.2, recall that even if the final goal is to perform classification and thus have a discrete output, *Boosting* has been shown to be an additive logistic regression model. In logistic regression, we build a function $f : \mathcal{X} \rightarrow \Re$ and estimate the probability $Pr[y = +1|x]$ of having $y = +1$ for a given test example x with $\sigma(f(x))$ where:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{5.2}$$

We compute f by maximizing the conditional likelihood of the data, or as is usually done in statistics, minimizing the negative log conditional likelihood instead, which turns to be:

$$\begin{aligned}\sum_j - \ln\left(\frac{1}{1 + e^{-f(x_j)}}\right) &= \sum_j - \ln(1) + \ln(1 + e^{-y_j f(x_j)}) \\ &= \sum_j \ln(1 + e^{-y_j f(x_j)}) \end{aligned} \tag{5.3}$$

Note that Equation 5.3 simplifies the notation compared to Chapter 4 by assuming $y_j = \pm 1$ as being the label of the example x_j (this is allowed since we describe the case of a binary classification). The minimum of the function in Equation 5.3 is not trivial to find and several methods have been developed to approximate it. In this work, we use the Newton-Raphson method, whose basic principles are explained in Appendix A. For now we assume that the minimum of the function in Equation 5.3 can be found and concentrate on how to use logistic regression to perform adaptation.

In the version showed in Algorithm 1, AdaBoost implicitly minimizes the *exponential* loss function of Equation 5.4 by minimizing Equation 5.5. The expression of the weights $D_t(j)$ before normalization is given in Equation 5.6.

$$\sum_j e^{(-y_j f(x_j))} \tag{5.4}$$

$$\sum_j D_t(j) e^{-y_j f(x_j)} \tag{5.5}$$

$$D_t(j) \approx e^{-y_j f_{t-1}(x_j)} \tag{5.6}$$

$$f_t(x) = \sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x) \tag{5.7}$$

Thus to apply logistic regression to *Boosting*, we have to modify the loss function of Equation 5.4 so that it matches the shape of the logistic regression function. This is done by leaving the final loss function in Equation 5.5 unchanged but modifying the weights $D_t(j)$ in the following way (still before normalization):

$$D_t(j) \approx \frac{1}{1 + e^{y_j f_{t-1}(x_j)}} \tag{5.8}$$

Thus, Equation 5.5 now becomes:

$$\begin{aligned}\sum_j D_t(j) e^{-y_j f(x_j)} &= \sum_j \frac{1}{1 + e^{y_j f_{t-1}(x_j)}} \cdot e^{-y_j h_t(x_j)} \\ &= \sum_j \frac{e^{-y_j h_t(x_j)}}{1 + e^{y_j f_{t-1}(x_j)}} \\ &= \sum_j \frac{1}{1 + e^{y_j f_{t-1}(x_j)}} \end{aligned} \tag{5.9}$$

which corresponds to the logistic function described by Equation 5.2.

We now explain how to include prior knowledge to the logistic loss function that we just described. We assume that we are given a prior model with a distribution $\pi(\cdot|x_j)$ for each example x_j and some training data d. The model that we want to build should fit both the prior model and the training data. We deal with this constraint by changing the logistic loss function of Equation 5.3 in a two-terms loss function. The first term measures the fit of the constructed model to the data d by using the log conditional likelihood, as before. The goal of the second term is to prevent the model built on the data to diverge from the prior model. The divergence between the two models is measured according to the binary relative entropy (also called Kullback-Leibler divergence and therefore abbreviated KL) between the prior model and the model built by *Boosting* on the training data plays this role. Formally, we have

$$\begin{aligned} &\sum_j \left[\ln\left(1 + e^{-y_j f(x_j)} + \eta KL\left(\pi_+(x_j) \| \sigma(f(x_j))\right)\right)\right] \\ = &\sum_j \left[\ln\left(1 + e^{-y_j f(x_j)} + \eta\left(\pi_+(x_j)\ln\left(1 + e^{-f(x_j)}\right)\right) + \eta\left(1 - \pi_+(x_j)\ln\left(1 + e^{f(x_j)}\right)\right)\right)\right] \end{aligned} \tag{5.10}$$

where

$$KL\left(p\|q\right) = p\ln\left(\frac{p}{q} + (1-p)\ln\left(\frac{1-p}{1-q}\right)\right)$$

and $\pi_+(x) = \pi(y = +1|x)$. The KL term basically measures the distance between two probability distributions. If the two distributions are independent, this term will have a large value. If on the contrary the two distributions are identical, the term will turn out to be null and the function will then be reduced to the first term only which is the normal *Boosting* case. In our case, we measure the distance between the distribution of the prior model built on the out-of-domain data $P(Y_j[l] = 1|x_j)$ and the distribution of the new model built on the in-domain data $\rho(f(x_j, l))$, where $\rho(x)$ is the logistic function $1/(1 + e^{-x})$. The weight η balances the influence of the logistic loss function and the binary relative entropy and is optimized using the held-out set.

5.5. Discussion of the Presented Methods

The methods presented above all combine two sets of data to build a classifier, but they differ in the way the combination of the data is done. Indeed, the combination of the two sets of data can be implemented at different levels:

- the data level (e.g. data concatenation)
- the feature or classifier level (e.g. *Boosting* adaptation)
- the classifier output or score level (e.g. logistic regression)

Despite the differences in the level at which the combination is done, note that these different implementations of adaptation *are* equivalent under certain conditions. For example, while one can perform the combination at the score level by interpolating the outputs obtained by two models, the same effect can be represented in a single interpolated model, i.e. at the classifier level, as typically done in language models (Stolcke, 2002). Similarly, data concatenation (data level) can be seen as an unweighted linear interpolation in certain cases.

Chapter 3 presented the MAP framework and some of its applications in the speech domain. Among the methods that were presented above, only the *Boosting* adaptation purely follows the MAP framework. In fact, *Boosting* adaptation is the only adaptation method that builds one model on the out-of-domain data and adapts the *same* model to the in-domain. Indeed, for data concatenation, logistic regression and the "feature" methods, the in-domain model does not use the MLE estimation of the parameters on the out-of-domain as the prior distribution of the parameters.

Considering the adapted model as being between the in-domain and the out-of-domain model, it should be noticed that the models created by the above-presented methods have different degree of similarity with the the in-domain and the out-of-domain model. For example, the similarity of the data concatenation model with the in-domain data can be seen as proportional to the percentage of in-domain data in the entire training data set, (the concatenation of the out-of-domain data and the in-domain data). For the other methods, the similarity with the in-domain and out-of-domain models is a function of the method parameters. Thus for logistic regression, the weights b_1, b_2, b_3 allow the final prediction to give more importance to one of the models, while for the *Boosting* adaptation, the same role is played by the weight η. Logistic regression and *Boosting* adaptation therefore have a higher freedom than data concatenation to follow the in-domain or the out-of-domain model.

This is not the case of the "feature" method, which is more closely related to the in-domain model. Indeed, for this particular method, the prediction of the out-of-domain model is represented only as one feature, while the in-domain model is represented by all of its features. The weight given to the out-of-domain model thus depends on the weighted number of times that the out-of-domain prediction feature is used by the base learner in the construction of the classifier. To give a highest weight to the out-of-domain model, the weighted number of times that the single out-of-domain prediction is used should be higher than the weighted number of times that all features of the in-domain are used, which is unlikely to happen. The "feature" method is thus meant to stay relatively close to the in-domain model.

CHAPTER

6 Data and Metrics

In this chapter, we present qualitatively and quantitatively the three corpora that we use for the meetings, telephone conversations and broadcast news speech styles (genres). We then detail the practical set-up of our experiments and describe the metrics that we use to measure the performance of sentence segmentation.

6.1. Data

The experiments presented in this work use three different corpora of data, that correspond to three different speech styles:

- meetings
- broadcast news
- telephone conversations

A qualitative description of these three corpora is presented in more detail below, while the main statistical characteristics of the data sets are shown in Table 6.3.

6.1.1 Meetings (MRDA)

The meeting data used come from the ICSI meeting corpus (MRDA) (Janin *et al.*, 2004). This corpus contains 75 meetings which are split into three main subtypes: “Even Deeper Understanding weekly meeting” (BED), “Robustness weekly meeting” (BMR) and “Meeting Recorder weekly meeting” (BRO). These three types of meetings have been recorded at the same place. The same settings were used in almost all meetings, especially the one headset microphone per participant. The meetings are however different in their substance since the people who took part in them are not the same and the subjects that they discuss also differ from one meeting type to another. The difference in the number of participants and the “relationships” between the participants are especially important. To illustrate this, imagine a meeting in which one of the participant is the chief. Further imagine that this chief turns out to be very authoritative and speaks most of the time with nobody daring interrupt him. In such a case, disruptions will surely not appear as often as in a meeting where people feel free to interrupt each other. Some statistics about the 3 types of meetings BED, BMR and BRO of the MRDA corpus are shown in Figure 6.1 and in Table 6.4.

The MRDA corpus has been detailed in Dhillon *et al.* (2004). We used transcriptions in which one *dialog act* has been attributed to each word boundary: if a dialog act unit (sentence unit) is ending with a word, the tag corresponding to the dialog act unit is attributed to the word with which the dialog act unit ends. The words on which no dialog act unit ends are assigned the tag 'n', for non-sentence unit. Statistics of the dialog acts frequency in the MRDA corpus are shown in Figure 6.2. The dialog acts are mapped onto the sentence boundary class (C_s) or the non-sentence boundary class (C_n). The backchannels, disruptions, floor grabbers, questions, statements, and the samples that were not annotated by the labelers are mapped onto the sentence boundary class. The rest of the instances are mapped onto the non-sentence boundary. Table 6.1 lists the dialog acts and their corresponding mapping for the MRDA corpus, but also for the SWBD and the TDT4 corpora that are presented below.

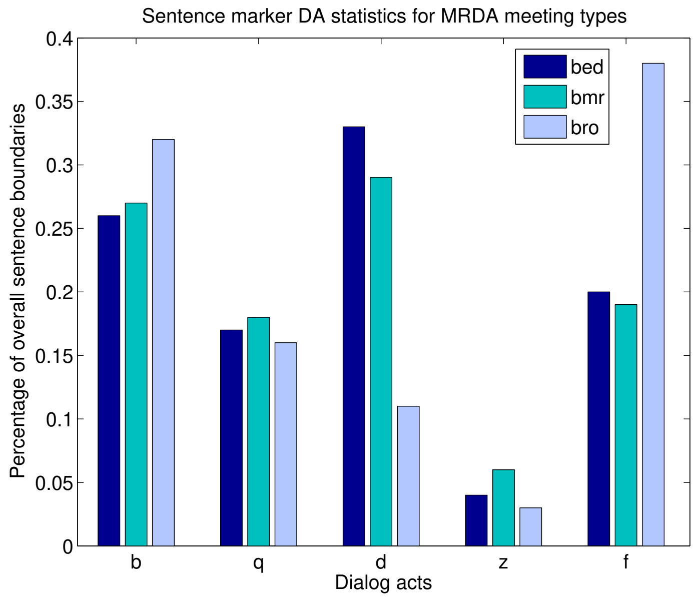


Figure 6.1: Repartition of the sentence boundaries between the 5 dialog acts. Note that the regular statement class 's' has been excluded from the statistics computation.

	Mapped on	MRDA	SWBD	TDT4
Backchannel	s	b	b	NA
Disruption	s	d	i	NA
Floor grabber	s	f	NA	NA
Question	s	q	q	NA
Statement	s	s	s	s
Unannotated	s	z	z	NA
Non SU	n	n	n	n

Table 6.1: Dialog acts details for each corpus. 'NA' means that the dialog act was not labeled in the corpus. In the column showing the mapping of the dialog act on the sentence boundary class or the non-sentence boundary class, 's' stands for the sentence boundary class, while 'n' stands for the non-sentence boundary class.

6.1.2 Telephone Conversations (SWBD)

The telephone conversations corpus that were used are taken from the subset of the Switchboard (SWBD) corpus provided by the LDC (RT04). It is the smallest of the three corpora in terms of the number of words. Figure 6.3 shows the frequency of each dialog acts in this corpus. Note that, as pointed out by Table 6.1, the dialog acts for SWBD are not identical to the ones presented above for the MRDA corpus. This is caused by the differences in the labeling guidelines. For example in the SWBD corpus, interruptions and floor grabbers were not marked, whereas they were in the MRDA corpus. However, even if the floor grabbers were not marked in the SWBD corpus, this does not affect the sentence segmentation task since the floor grabbers were implicitly marked as incompletes ('i'), and the incompletes are mapped to the same class 's' as the floor grabbers. For more details about the labeling differences between MRDA and SWBD, one can refer to Dhillon *et al.* (2004).

6.1.3 Broadcast News (TDT4)

The broadcast news corpus is the largest of the three corpora in terms of the number of words. However, even if the absolute training data is bigger than for MRDA and SWBD, it should be noted that the number of sentence boundaries is approximately the same in TDT4 and MRDA. Indeed, while telephone conversations and meetings are both conversational speech, broadcast news imply long segments with only one speaker and thus less interruptions in the speech. Hence, the average sentence unit length is approximately twice as big on the TDT4 corpus than on the MRDA as shown in Table 6.3. The speech style differences between conversational speech and broadcast news also appear in the dialog acts. Indeed, the broadcast news shows of TDT4 have been labeled with only two dialog act tags, the sentence boundary one ('s') and the non-sentence boundary one ('n'). This is reasonable, since there usually is only one simultaneous speaker and therefore fewer overlaps between the speakers. Indeed, the interaction between the main speaker and the reporters on the field are usually a question-answer based conversation and do not lead to much disruptions or backchannels.

6.1.4 Accuracy of the Labeling

Although the human-labeling is considered as true during the classification task, it is important to note that the labeling can be noisy. Indeed, the labeling is done by different labelers which have different sensitivities and ways of interpreting the labeling guidelines. Therefore, although the guidelines are supposed to lead to a consistent labeling, this is not always the case. One way of measuring the inaccuracy in the labeling of some data is to make different person label the same data and measure the *pairwise human agreement*. Unfortunately we could not find the sentence boundary pairwise human agreement neither for the MRDA corpus nor for the SWBD one (the TDT4 has no such number since the sentence boundaries have been inserted according to their original textual transcriptions). Although there are no quantitative measures, a qualitative estimation seems to show that the SWBD corpus contains some labeling inconsistencies. For example, a triple repetition of the word "`yeah`" is sometimes labeled as three different sentence units "`yeah. yeah. yeah.`" and sometimes as only one "`yeah yeah yeah`". We did not find such errors in the MRDA corpus for which the guidelines given to the labelers were more precise.[1]

[1] Thank you to Dan Jurafsky and Elizabeth Shriberg for their helpful explanations on the labeling process of the SWBD and the MRDA corpus.

6.2. Practical Set-Up

Both the MRDA corpus and the TDT4 corpus are available in the reference and STT conditions described in Subsection 2.2.3. For the SWBD corpus, the ASR output of the telephone conversations was not available no forced alignment could thus be performed to obtain the files in STT conditions.

From a performance viewpoint, the STT transcriptions incorporate the errors made by the ASR in the process of recognizing the words (as an order of magnitude the word error rate on the MRDA corpus is 35.4% (Jurafsky *et al.*, 1998)) and the classifier performance is therefore expected to be worse on them than on the reference transcriptions. The study under the STT conditions is however of big interest in the effort of reducing the human work, since it allows one to get transcriptions of meetings automatically and requires human work only for the dialog act labeling task.

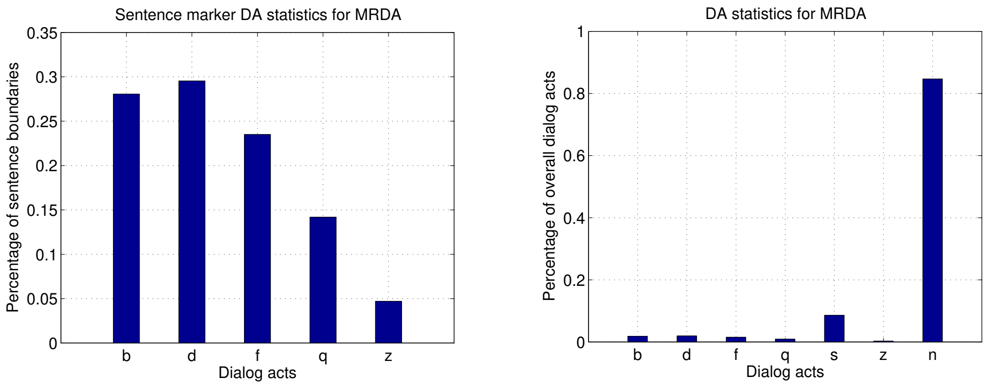
In both reference and STT conditions the transcriptions contain various information. An excerpt from one meeting transcription is shown in Table 6.2. Although this is a transcription from a meeting of the MRDA corpus, the files from all three corpora are in the same format.

a	288.170	0.110	the	fe004	n
a	288.280	0.470	students	fe004	n
a	288.750	0.120	in	fe004	n
a	288.870	0.110	the	fe004	n
a	288.980	0.360	undergrad	fe004	n
a	289.340	0.380	class	fe004	n
a	289.720	0.090	that	fe004	n
a	289.810	0.150	he's	fe004	n
a	289.960	0.370	teaching	fe004	q
2	290.770	0.540	um	mn015	f
2	291.490	0.260	well	mn015	n
2	291.750	0.190	he	mn015	n
a	291.857	0.340	e-	fe004	d
2	291.940	0.380	said	mn015	n
2	292.850	0.640	um	mn015	n

Table 6.2: An excerpt from a meeting transcription.

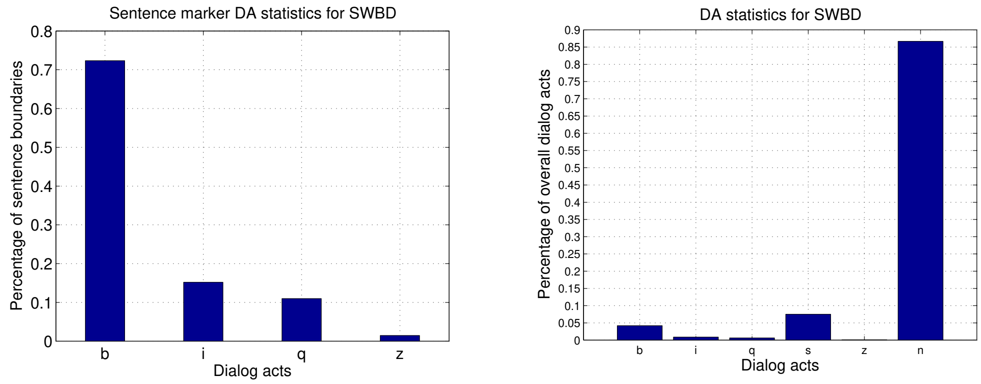
The first column is the channel identifier and is correlated with the fifth column which identifies the speaker. The next two columns show the time at which the word on column four begun and how long it last as estimated by the ASR. The last column contains the label that indicates which dialog act follows the word in column four. The letters 'b', 'd', 'f', 'q', 's' stand for the five types of dialog acts described in Subsection 2.1.2 and summarized in Table 6.1. Using the mapping of Table 6.1, one can thus say if the word is followed by a sentence boundary or not.

Table 6.3 provides numerical comparisons of the MRDA, TDT4 and SWBD corpora. For MRDA and TDT4, although the meetings are the same for the reference and the STT conditions, the size of the corpus in STT conditions is not the same as the one in reference conditions. The lack of data in STT conditions appears because the ASR was unable to recognize certain portions of the speech from the audio. For example, if the speaker talks quietly or if there is a lot of ambient noise, the ASR will not be able to extract words from the audio files and entire parts of the audio wave will be rejected. The smaller size of the file logically reflects on the vocabulary size and the STT vocabularies contain less words than the reference ones.



(a) Sentence marker dialog acts (b) All dialog acts

Figure 6.2: Dialog acts statistics for the MRDA corpus; 's' stands for sentence boundary, 'n' for non-sentence boundary, 'b' for backchannel, 'd' for disruption, 'f' for floor grabber, 'q' for question and 'z' for the unannotated dialog acts. The graph on the left shows the relative percentage of the dialog acts that corresponds to the sentence boundary class, except the dialog act 's'. The graph on the right shows frequencies of all dialog act classes.



(a) Sentence marker dialog acts (b) All dialog acts

Figure 6.3: Dialog acts statistics for the SWBD corpus; 's' stands for sentence boundary, 'n' for non-sentence boundary, 'b' for backchannel, 'i' for incomplete sentence and 'q' for question. The graph on the left shows the relative percentage of the dialog acts that corresponds to the sentence boundary class, except the dialog act 's'. The graph on the right shows frequencies of all dialog act classes.

	MRDA		**SWBD**	**TDT4**	
	reference	STT	reference	reference	STT
Training set size	538,956	476,041	379,498	1,134,860	1,122,976
Test set size	101,510	87,576	-	84,787	82,644
Held-out set size	110,851	94,433	-	83,760	81,788
Vocabulary Size	11,034	12,092	13,109	32,209	25,915
Average utterance length	6.54	7.65	7.57	14.69	14.59

Table 6.3: Numerical data characteristics for MRDA and SWBD. Sizes and sets are given in number of words.

	BED	BRO	BMR
Number of meetings	15	23	29
Total size	144,554	198,618	317,949
Average size of one meeting	9,637	8,636	10,963
Average speaker number	5.67	5.74	6.721
Average number of words between speaker changes	8.11	8.92	5.50
Average utterance length	6.56	6.38	6.71
Vocabulary Size	4,467	4,826	6,681

Table 6.4: Numerical data characteristics for the three MRDA meeting types; sizes and sets are given in number of words.

6.3. Features

The features used throughout the experiments are listed in Table 6.5. They can be split into three main classes: the lexical features, the prosodic features and a mix of both of them. In the presentation of the experiments in chapters 7 and 7.5, the four following sets of features will be distinguished, which contain the following features:

- lexical features only (*lexical*),
- lexical features and the pause duration, including the discretized pause duration (*lexical+*),
- prosodic features only, including the pause duration and various pitch and energy measures (*prosodic*),
- all lexical and prosodic features (*all*).

6.3.1 Lexical Features

The decision of hypothesizing a sentence boundary or not has to be made after each word. It is thus natural to look for features that relate to the word. That is what is done for the lexical features, considering N-grams of different sizes that combine the word after which the decision is made (which is designated as *current word*) and the two words that surround it, i.e. the *previous word* and the *next word*. The length of the N-grams is limited to three (trigrams), since in our experiments longer N-grams did not bring significant improvements in the segmentation performance and they make the training task much slower, as the number of features increases significantly.

6.3.2 Prosodic Features

For the prosodic features, we distinguish two pause durations: the *intra-channel* pause duration, which is independent for each channel, and the *inter-channel* pause duration, which stops timing a pause whenever a word is uttered on any channel. The intra-channel pause duration is useful since it depicts a cleaner picture of the pauses than the inter-channel one. In particular the intra-channel pause duration does not take backchannels into account as opposed to the inter-channel pause duration. The inter-channel pause duration is nevertheless also useful since it can help to detect disruptions, which are often signaled by a null pause or even more often an overlap. In this work, no overlap information has been included in the feature set, but it could be worth to try it according to Cetin & Shriberg (2006).

The other prosodic features used are all related to the pitch and energy in the voice of the speaker. More precisely, they are F0 derived features which take various measures between two adjacent windows or two adjacent words. Examples of these measures are the logarithmic ratio between the minimum, the maximum and the mean energy levels in two adjacent windows (resp. words), or the slope of the pitch and its logarithmic ratio between two windows (resp. words). They were extracted with a Java-based tool called Algemy developed at the Stanford Research Institute (SRI) (Bratt, 2006). In total, the prosodic features set contains 34 features.

6.3.3 Combination of Lexical and Prosodic Features

The combination of the prosodic and lexical features is shaped only by two features, one prosodic and one lexical. It uses the intra-channel pause duration between two words and combines it with the two words. The motivation behind this is that neither the bigram composed of the two words or the pause duration can achieve an accurate classification by themselves in some cases. For example, consider the following two sentences:

```
''This is the fork I'm gonna eat with. It is much better than a
  regular fork!''
```

At the separation point between the two sentences, i.e. where the sentence boundary is, the word bigram is "`with it`". This particular bigram is not a relevant indicator of a sentence boundary between `with` and `it`, since it is much more usual to see it within a sentence than as it appears here. Therefore the use of lexical features would most likely lead to hypothesize a non-sentence boundary between "`with`" and "`it`". Prosodic features only do not ensure either to detect correctly this particular sentence boundary, since the speaker may have not used a proper intonation. It is however likely that the speaker made a pause in his speech between the words "`with`" and "`it`". We can use this pause by performing a binary discretization on it and creating a new trigram `(with, 1, it)` if the pause is higher than a given threshold and `(with, 0, it)` if the pause is lower than the threshold. The threshold value was set to 0.2 seconds, which empirically revealed to be the most performing one.

Lexical	**Prosodic**	**Lexical and prosodic**
current word	intra-channel pause duration	(curr., discret. pause)
previous word	inter-channel pause duration	(curr., discret. pause, next)
next word	energy ratio/log. ratio	
(current word, next word)	pitch ratio/log. ratio	
(prev. word, curr. w., next w.)		

Table 6.5: List of features used to describe an instance to the classifier. For the lexical features, "current word" stands for the word after which the decision of hypothesizing a sentence boundary or a non-sentence boundary has to be made. The "previous" and "next" word are the words surrounding it in the sequence of words.

6.4. Method

To build a classifier, the learning algorithm described in Algorithm 1 is provided with the instances of the training set; each instance is a vector of the feature space. There are four different feature spaces which correspond to the four different sets of features described in the previous section, but for a given experiment, all instances must be described in the same feature space. After T iterations, *Boosting* outputs a function $f(x)$ which computes the probability $p(C_s|x)$ that the instance x is hypothesized as a sentence boundary. An *acceptance threshold* t_{acc} is then used to map this probability on either a sentence boundary or a non-sentence boundary, and thus decide of the class $C(x)$ of the instance x:

$$C(x) = \begin{cases} C_s & \text{if } p(C_s|x) > t_{acc} \\ C_n & \text{otherwise} \end{cases} \tag{6.1}$$

where C_s stands for the sentence boundary class and C_n the non-sentence boundary class.

The acceptance threshold t_{acc} is a parameter that can be optimized on the held-out set; its value is by default set to 0.5. As will be shown further in the practical results, a lower acceptance threshold leads the final hypothesis to contain more sentence boundaries, while a higher acceptance threshold on the contrary leads to less hypothesized sentence boundaries. Varying t_{acc} can help adjusting the number of hypothesized sentence units so that it matches with the sentence unit frequency of the domain to which the classifier is applied. It can also be used to give a precise profile to the classifier by balancing the performance between the recall and the precision, two metrics that are presented in the following section.

6.5. Metrics

To estimate the accuracy of a classification, we compare for each instance in the test set the class estimated by the classifier to the one assigned by humans, which we refer to as the correct one.[2] In the case of a binary classification task, the classification guessed by the classifier can be either the same or different as the correct one; moreover, the example that is classified belongs to either the sentence boundary class C_s or the non-sentence boundary class C_n.

This leads to four designations to compare the estimation of the system with the correct classification: the true positives/negatives and the false positives/negatives. The true positives and the true negatives represent the examples that the classifier guessed correctly. More precisely, a sentence boundary that was identified as a sentence boundary by the classifier is a *true positive* whereas a non-sentence boundary that the classifier identified as a non-sentence boundary is a *true negative*. The true negatives are much more frequent, since the non-sentence boundaries are more frequent than the sentence boundaries. The two remaining classes are the ones containing the examples that were wrongly classified. The *false negatives* represent the sentence boundary that the classifier missed, that is to say the sentence boundaries that the classifier estimated as non-sentence boundaries. Finally, the *false positives* are the non-sentence boundaries that the classifier estimated as sentence boundaries. The designations are summarized in Figure 6.4.

To measure the performance of a classification as a concrete number, we used the *F-measure* and the error defined by the National Institute of Standards and Technology (NIST), which we refer to as being the *NIST-SU* error rate. These two measures are both combinations of the four designations that were described in the previous paragraph.

6.5.1 F-Measure

In order to understand what the F-measure is, one first has to understand two additional concepts, the *recall* and the *precision*. The recall is the ratio of sentence boundaries hypothesized by the classifier to the total number of reference sentence boundaries. It can be expressed as:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{6.2}$$

[2] As already mentioned, even though humans can make mistakes in their labeling, there is no better way to approach the truth. The labeling done by humans is thus considered as the absolute truth when evaluating the hypotheses done by a classifier. In this work, the sentence boundaries assigned by human labelers will also be referred to as *true sentence boundaries* or *reference sentence boundaries*.

	Correct classification	System estimation	Abbreviation
True positive	SU	SU	TP
True negative	non-SU	non-SU	TN
False positive	non-SU	SU	FP
False negative	SU	non-SU	FN

Figure 6.4: The four possible combinations between the correct classification and the system estimation. "SU" stands for sentence boundary.

Hence, if only recall was taken into account to measure the performance, the best solution would be to hypothesize sentence boundaries everywhere, which would yield to a recall of 100%. Such a case is avoided by combining the recall measure with the precision measure. This latter is the ratio of true sentence boundaries within the hypothetized sentence boundaries to the overall number of hypothetized sentence boundaries:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.3}$$

In the previous extreme case with a recall of 100%, the precision would thus be very low. On the contrary, if only non-sentence boundaries are hypothesized the precision is 100%, but the recall is 0%. Formally, using the abbreviation of Figure 6.4, we have:

Figure 6.5 shows an example of a sequence of words with the reference sentence boundaries and the sentence boundaries estimated by the classifier. Let us compute the recall and the precision for this classification. For the recall, we need the number of true positives and the number of false negatives. As indicated in Figure 6.5, there are two true positives (#4 and #8) and two false negatives (#5 and #7). The recall is thus $2/(2+2) = 0.5$. For the precision, we need again the count of true positives (2) and the count of false positives, which is only 1 (#1). Hence, the precision for this example is $2/(2+1) = 0.66$.

The recall and precision measures taken separately are however not sufficient, as pointed out by the above-mentioned extreme cases. These two measures are complementary and should therefore be combined. This is achieved by the F-measure, which is nothing but the weighted harmonic mean of these recall (R) and precision (P) measures:

$$\text{F-Measure} = \frac{P \cdot R(\alpha + 1)}{\alpha P + R}$$

where α is a coefficient usually set to 1.

Note that even if recall and precision are not used as evaluation means, it can be worth looking at their behavior in some cases. For instance, it could be easier for further speech processing tasks, such as summarization, to have very short sentence units, as opposed to very long sentence units. In this case, we would like to build classifiers that rather find more sentence boundaries than not enough and therefore aim at having a high recall, while precision would become less important.

6.5.2 NIST-SU Error Rate

Beyond F-Measure, which is a widely used measure for classification performance evaluation, we use a second more field related metric, the NIST-SU error rate. The NIST-SU error rate is the ratio of the number of wrong hypotheses made by the classifier to the number of reference sentence boundaries. Expressed in terms of the four designations of Figure 6.4, this is:

$$\text{NIST-SU} = \frac{FN + FP}{TP + FN}.$$

Reference	`at.the.same.time|no|oh.okay|no|`
System	`at|the.same.time|no.oh.okay.no|`
Errors	`E C C C E C E C`
Classification	`FP TN TN TP FN TN FN TP`
SU-id	`1 2 3 4 5 6 7 8`

Figure 6.5: The first two lines show the reference sentence segmentation and the system estimate for the sequence of words "`at the same time no oh okay no`". Sentence boundaries are marked with a '|' and non-sentence boundaries with a '.'. The third line indicates if the system guessed correctly (C) or made an error (E). The fourth line indicates the type of classification between true positive (TP), true negative (TN), false positive (FP) and false negative (FN). The last line shows the ID attributed to each classified example, for reference purposes.

Metric	Computation	Value
NIST-SU	$\frac{\mathbf{2}(FN)+\mathbf{1}(FP)}{\mathbf{2}(TP)+\mathbf{2}(FN)}$	0.75
Recall	$\frac{\mathbf{2}(TP)}{\mathbf{3}(TP)+\mathbf{1}(FN)}$	0.50
Precision	$\frac{\mathbf{2}(TP)}{\mathbf{2}(TP)+\mathbf{1}(FP)}$	0.66
F-Measure ($\alpha = 1$)	$\frac{\mathbf{2}(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}$	0.80
Test error	$\frac{\mathbf{2}(FN)+\mathbf{1}(FP)}{\mathbf{3}(TN)+\mathbf{2}(TP)+\mathbf{1}(FP)+\mathbf{2}(FN)}$	0.38

So if no boundaries are marked by the classifier, the NIST-SU error rate is 100%, but it can exceed 100% in some cases. The maximum NIST-SU error rate is indeed the ratio of number of words to the number of reference sentence boundaries. In the example of Figure 6.5, we have two false negatives (#5 and #7), one false positive (#1) and two true positives (#4 and #8), which yields a NIST-SU error rate of $(2+1)/2+2 = 0.75$. Note that while the NIST-SU is an error rate measure and thus tends to decrease when the classification becomes more accurate, the F-Measure has the opposed behavior and increases with more accurate classifications.

6.5.3 Test Error Rate

Although we will rarely use it to evaluate the performance of a classification, we will sometimes refer to the *test error* as being the number of errors done in the classification, divided by the total number of instances on which the classifier was tested:

$$\text{Test error} = \frac{FN + FP}{FN + FP + TP + TN}.$$

In the case of the example of Figure 6.5, we have $3/80 = 0.38$.

CHAPTER 7

Experiments and Results

Experimental results are presented in this chapter. The first experiment presents a cross-classification among MRDA, SWBD and TDT4, showing the need for adaptation among these speech styles. The second set of experiments describes adaptation of the conversational speech style data from the SWBD corpus to the meeting speech style of the MRDA corpus using the four methods presented in Chapter 5. Adaptation is performed on the entire MRDA corpus and on a subpart of it. The third part of the chapter studies in more detail the importance of the features.

7.1. Experiment Set-Up

We used the split of data usually used in learning tasks and describe in the previous chapter, that is to say a training set, a test set and a development (or held-out) set. The training sets contains approximately 80% of the labeled data whereas the remaining 20% are evenly split between the test set and the development. For the MRDA and the TDT4 corpora, we used the same splits of the data as in Ang *et al.* (2005) and Zimmermann *et al.* (2006b) respectively. The training of the classifier is done on the training set, the held-out set is optionally used to optimize parameters and the test set is used to evaluate the performance of the classifier.

When nothing is mentioned, the data are represented by the *lexical*+ features set, that is to say lexical features and the pause duration. The choice of the *lexical*+ features set and its 9 features is a trade-off between the learning time and the accuracy of the classifier.

7.2. Cross-Classification among MRDA, SWBD and TDT4

The statistics of the three corpora that we are dealing with are shown in Table 6.3. The first experiment that we do to see how the corpora can perform on each other is to perform a basic cross-classification experiment. This experiment consists of building one classifier on each of the three training sets and assess their performance on the test set of each of the three corpora. The NIST-SU error rates of this experiment are shown in Table 7.1.

First, we observe that each classifier built on a specific corpus is the best performer when evaluated on this very same corpus. This is an expected result since in these particular cases both the data on which the classifier is built and the data on which the classifier is tested are of the same speech style and share the same settings (recording, labeling, etc.). Another observation is that classification on the STT conditions leads to higher error rates than on the reference conditions. This is, as explained in Section 6.1, because the classification is done on top of the ASR output which already contains errors, and the data are thus less consistent.

A different behavior is however to be noticed for MRDA and TDT4 on what relates to the difference between reference and STT conditions. Indeed, while the MRDA reference and STT classifiers perform approximately the same (same on STT and 1.6% absolute difference on reference), the TDT4 reference classifier performs 14.3% absolute better on TDT4 in reference conditions than the TDT4-STT classifier, while the inverse situation shows a difference of only 6.5% absolute in favor of the STT classifier. These differences between the TDT4-REF and the TDT4-STT classifiers can be explained by the large number of words available in the TDT4 corpus which is likely to lead to more word recognition errors than on the MRDA corpus, where the words that have a strong correlation with the sentence boundary are easier to detect (more on this topic in Section 7.5). Therefore, the TDT4-REF classifier, which was trained on the true and thus reliable sequence of words, is particularly penalized when evaluated on STT conditions.

7.2.1 Corpora Ordering According to the Difficulty of Classification

Let us compare, for each classifier built on a corpus C, the classification performance obtained on the test set of the same corpus C: for MRDA, the NIST-SU error rate is 37.2%; for SWBD, the NIST error rate is 48.5% while it is 56.1% for TDT4. These numbers tend to show an ordering in the difficulty of the classification task for each corpus. It is especially outstanding that TDT4 has the worst error rate, while its speech properties (one speaker, prepared text, professional reader, etc.) are supposed to make it ideal for the sentence segmentation task. Moreover, an eventual lack of training data with respect to MRDA or SWBD is not a valid argument since the TDT4 corpus is the largest one among the three corpora. Hence, the highest error rate for TDT4 has another source, which is the low sentence units frequency (14.6 words per sentence vs. 6.5 and 7.6 for MRDA and SWBD respectively), i.e. the sentence units in the TDT4 corpus are longer.

	Test corpus				
Training corpus	**TDT4-REF**	**TDT4-STT**	**MRDA-REF**	**MRDA-STT**	**SWBD-REF**
TDT4-REF	**56.1%**	82.9%	66.4%	74.7%	90.4%
TDT4-STT	70.4%	**76.4%**	71.2%	72.6%	91.5%
MRDA-REF	95.1%	120.5%	**37.2%**	50.2%	71.0%
MRDA-STT	100.2%	106.1%	38.8%	**50.1%**	72.4%
SWBD-REF	97.2%	129.0%	45.7%	60.0%	**48.5%**

Table 7.1: NIST-SU error rates for the basic cross-classification experiment with the default acceptance threshold set to 0.5.

This low sentence units makes it penalizing with the NIST-SU error rate, since the divisor in the NIST-SU formula is the number of sentence units. Using the test error instead of the NIST-SU error would result in having TDT4 performing the best, since the divisor would in this case be the total number of words. This explanation does however not hold for SWBD and the higher error rate compared to MRDA is more likely to be explained by inconsistencies in the labeling. Indeed, as explained in Section 6.1, even if there are no human agreement number for the sentence boundary labeling for the SWBD and the MRDA corpora, it is a fact that labelers had more precise instructions on how to label the MRDA corpus than for the SWBD corpus. Furthermore, as the MRDA corpus was created after the SWBD corpus, the experience of the SWBD annotation must also have been useful to improve the consistency of the labeling.

The labeling inconsistencies do yet not prevent the SWBD classifier to do very well when evaluated on MRDA in reference conditions (MRDA-REF). It only performs 8.5% absolute worse than the MRDA-REF classifier does in this case (45.7% vs. 37.2%). One should notice that this is even 2.8% absolute better than the performance of the SWBD classifier when evaluated on SWBD itself, which tops 48.5%. This observation confirms that the SWBD corpus seems to be more difficult to classify than the MRDA-REF one. This efficiency of the SWBD classifier on MRDA-REF is all the more noticeable when compared to the poor performance obtained by the TDT4-REF classifier on MRDA-REF (66.4%). This difference between the performance of the SWBD classifier and the TDT4-REF classifier evaluated on MRDA-REF points out the importance of the *speech style*.

Indeed, while these three corpora all represent three different speech styles, telephone conversations and meetings share some characteristics that TDT4 does not have. In particular, telephone conversations and meetings are both conversational speech style in a multi-speaker environment, which implies disruptions and hesitations in the utterances. This is typically not the case of broadcast news, since there is most of the time only one speaker who reads out loud a written text. This reduces the spontaneity in the speech and the probability of hesitations, language disfluencies and interruptions by other speakers. These high-level differences can also be noticed in the poor classification performance of SWBD and MRDA-REF classifiers evaluated on TDT4-REF (97.15% and 95.05% vs. 56.06% for TDT4).

7.2.2 Varying the Acceptance Threshold

As explained in Section 6.4, an acceptance threshold has to be chosen, in order for the probability $p(C_s|x)$ of having a sentence boundary for an instance x output by the classifier, to be converted into an actual class C. The class C is either the sentence boundary class C_s, or the non-sentence boundary one C_n. Table 7.2 shows the same result as Table 7.1 but with the acceptance threshold that optimizes the NIST-SU error rate on the test set.[1] Reading the acceptance threshold values that lay in the diagonal from the upper left to the lower right of the table shows that the optimal acceptance threshold for the probabilities output by the classifier of the corpus that is being evaluated (SWBD for SWBD, for example) is always 0.5. When on the contrary the corpus on which the classifier has been trained has a different speech style from the domain on which it is evaluated, the optimal acceptance threshold is usually not the same. An example of this is shown in the MRDA STT column, where both TDT4 classifiers have optimal acceptance thresholds of 0.3 while the two MRDA classifiers and the SWBD classifier have an optimal acceptance threshold of 0.5. These differences in the value of the acceptance threshold can be explained by the lower sentence unit frequency in the TDT4 corpus (one SU every 14.6 words) than in the SWBD and MRDA ones (6.5 and 7.6). As a consequence, the classifier has to compensate its lack of sentence unit hypotheses by lowering the acceptance threshold.

This explanation does however not hold in the inverted case in which we would expect the MRDA and SWBD classifiers to increase their acceptance threshold when they are evaluated on the TDT4 corpus, so that less sentence boundaries would be hypothesized. Table 7.2 shows that the MRDA-REF and the MRDA-STT classifiers have an acceptance threshold of 0.4 and 0.3 respectively, while they have an optimal acceptance threshold of 0.5 and 0.4 when evaluated on MRDA-REF. However, when the error rates are as high as they are in this case, that is just below 100% NIST-SU error rate, it does not seem to make much sense to try to draw any conclusion from them. Indeed, a 100% NIST-SU error rate is the error rate that a classifier hypothesizing

[1] We are aware that the approach used here would not be correct if our goal was to actually compare the NIST-SU error rate obtained with the optimized acceptance threshold to the one of the standard one and make any deduction from the acceptance threshold optimization. In such a case, we would need to optimize the acceptance threshold on the held-out set for each experiment and then use this acceptance threshold as the optimal one, although it might actually not be the optimal one on the test set. But because our goal is here to show that the actual optimal acceptance threshold on the test set is different from the default one, we do not use this standard approach.

only non-sentence boundaries (called chance classifier) would reach and it might be that the results around this number lead to hypotheses that are as wrong as the ones done by the chance classifier. This is typically the case for large acceptance thresholds, such as for example the 0.9 acceptance threshold for MRDA-REF evaluated on TDT4-STT.

	Test corpus				
Training corpus	**TDT4-REF**	**TDT4-STT**	**MRDA-REF**	**MRDA-STT**	**SWBD**
TDT4-REF	**56.0% [0.5]**	81.1% [0.6]	63.5% [0.3]	73.1% [0.3]	89.9% [0.6]
TDT4-STT	69.9% [0.4]	**76.5% [0.5]**	64.4% [0.2]	71.1% [0.3]	91.5% [0.5]
MRDA-REF	94.8% [0.4]	98.7% [0.9]	**37.2% [0.5]**	50.1% [0.5]	70.8% [0.4]
MRDA-STT	94.9% [0.3]	99.3% [0.9]	37.9% [0.4]	**50.1% [0.5]**	71.2% [0.4]
SWBD	97.1% [0.5]	102.1% [0.9]	45.7% [0.5]	60.0% [0.5]	**48.5% [0.5]**

Table 7.2: NIST-SU error rates for the basic cross-classification experiment. Each pair of result contains the NIST-SU error rate for the optimal acceptance threshold, which is the number in brackets.

Figure 7.1 shows a typical recall vs. precision graph for 20 different acceptance thresholds going from 0.95 to 0.0 with a step of 0.05 between each of them. The values are the ones obtained by the MRDA reference classifier evaluated on the MRDA reference corpus. This graph shows the relationship between the recall and precision measures and the acceptance threshold. Varying the acceptance threshold allows one to balance the precision and the recall yielded by the classifier, which could be useful in some applications as explained in Section 6.5. The minimum acceptance threshold 0.0 (the point at the most right on the graph) has the effect of hypothesizing only sentence boundaries. The recall is thus 100%, whereas the precision is the number of true sentence boundaries divided by the total number of instances to classify. This is nothing but the inverse of the sentence unit frequency: $1/6.5 = 15.4\%$. The other extreme point, at the most left, is the one where the acceptance threshold tends to 1.0 (it cannot be 1.0 since the number of hypothesized sentence boundaries would then be null and the precision would have a null denominator). This means that very few sentence boundaries are hypothesized and the precision tends to 100%, while the recall is very low.

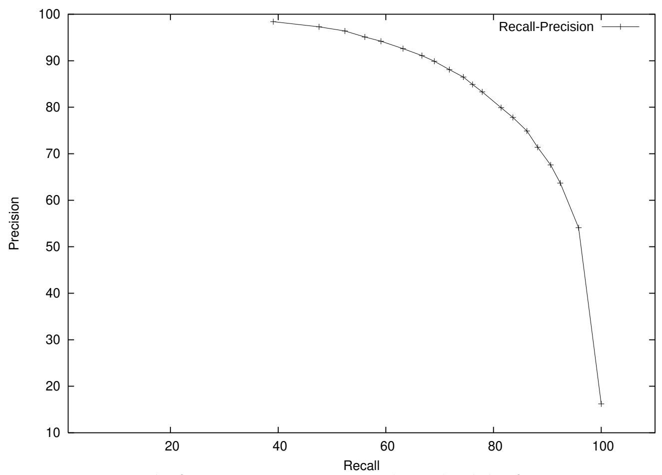


Figure 7.1: Recall-Precision graph for 20 acceptance thresholds from 0.95 to 0.0 with a gap of 0.05 between each of them.

We have seen the effect of varying the acceptance threshold on the NIST-SU error rate through Table 7.1 and Table 7.2. Figure 7.1 then showed us the behavior of the recall and the precision as functions of the acceptance threshold. The overall picture with the NIST-SU error rate, the recall, the precision, and the F-Measure as functions of the acceptance threshold shown in Figure 7.2 . We observe that the optimal acceptance threshold is well and truly 0.5, as already noticed from the results in Table 7.2, but that the performance obtained with the other acceptance thresholds is not very far from the optimal one. The maximum F-Measure value shows up on the point where the recall and the precision curves cross. As for the NIST-SU error rate, there is no obvious maximum for the F-Measure since the curve is pretty flat as its highest values, i.e. for acceptance thresholds value between 0.25 and 0.5. We can interpret this as being the zone where a loss in the precision is still compensate by a gain in the recall and vice versa.

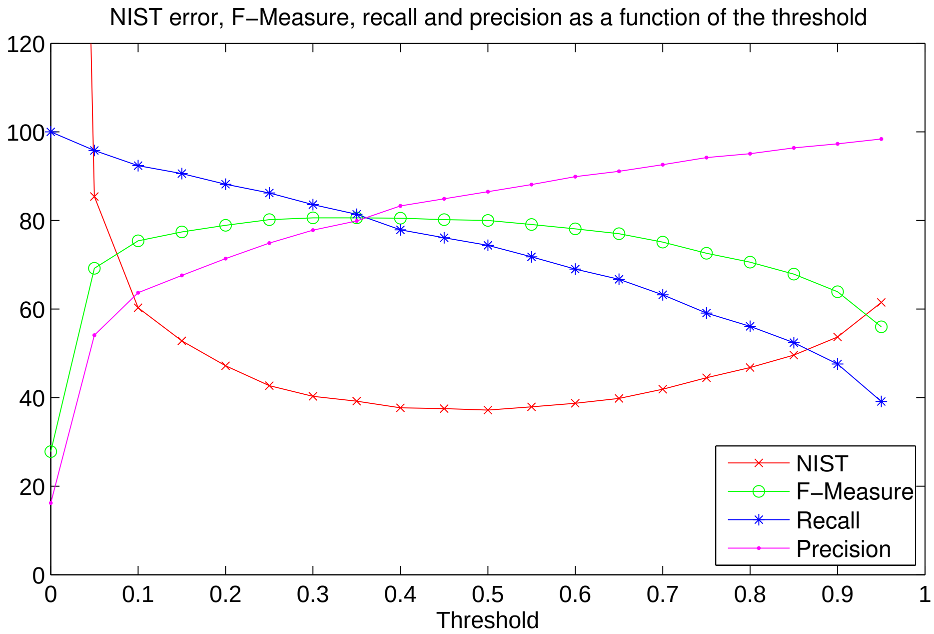


Figure 7.2: NIST-SU error rate, F-Measure, recall and precision as a function of the acceptance threshold; graph for the MRDA-REF classifier evaluated on the MRDA-REF test set.

7.3. Adaptation of SWBD on MRDA

Section 7.2 has given us a first insight into the characteristics of the three corpora that we are dealing with. In particular, we have observed that the speech style and the labeling are two major differences among those corpora. What we did in Section 7.2 is not properly speaking an adaptation, since we used a classifier trained on one domain (out-of-domain) to classify data from another domain (in-domain), but did not use any in-domain training data. More precisely, this can be considered as *unsupervised* classification since in the building phase of the classifier, we do not use any data from the domain on which we are testing the classifier. In the reality, it is however reasonable to assume that we can count on some labeled data from the in-domain, and thus use the approaches presented in Chapter 5 to perform *supervised* adaptation. As explained in Chapter 3, the two domains involved in the adaptation process should share some similarities. Since the SWBD and the MRDA corpora seemed to be the two corpora that are the most similar according to the first basic cross-classification presented in the previous section, we perform adaptation on these two domains, using the MRDA corpus as the in-domain and the SWBD corpus as the out-of-domain.

Learning curves

The next question is to know how to evaluate the adaptation methods presented in Chapter 5. The experiment done in the previous section was very basic since it always assumed no data from the in-domain and a very large set of data of the out-of-domain. To better match with the reality, we will study the effect of having different amount of the in-domain data which we represent on graphs with *learning curves*.

These learning curves are shown both for the whole MRDA corpus and for only one of its meeting types, the BED one. Performing an experiment with only the BED type of meetings aim at reducing the variance between the instances, which could be increased by the difference that appear between the different types of meetings, as explained in Section 6.1.

Note on the loss function used for all following experiments

Recall that the *Boosting* algorithm that we use gives us the choice between two loss functions: the exponential loss function and the logistic loss function. Figure 7.3 shows a comparison between the two of them for the MRDA classification. The logistic loss function performs slightly better and we use it for the rest of the experiments of this subsection.

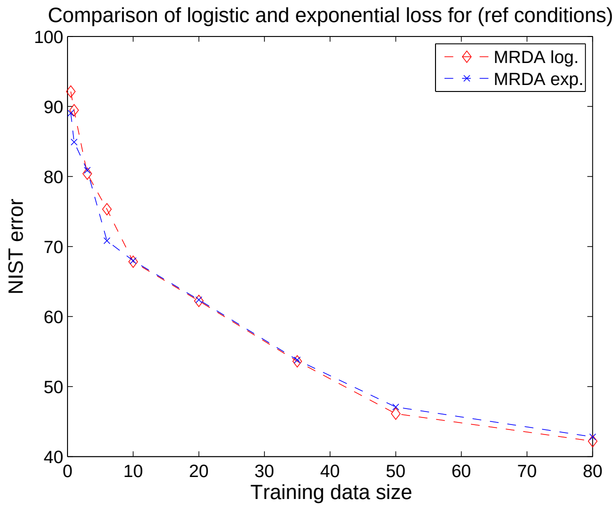
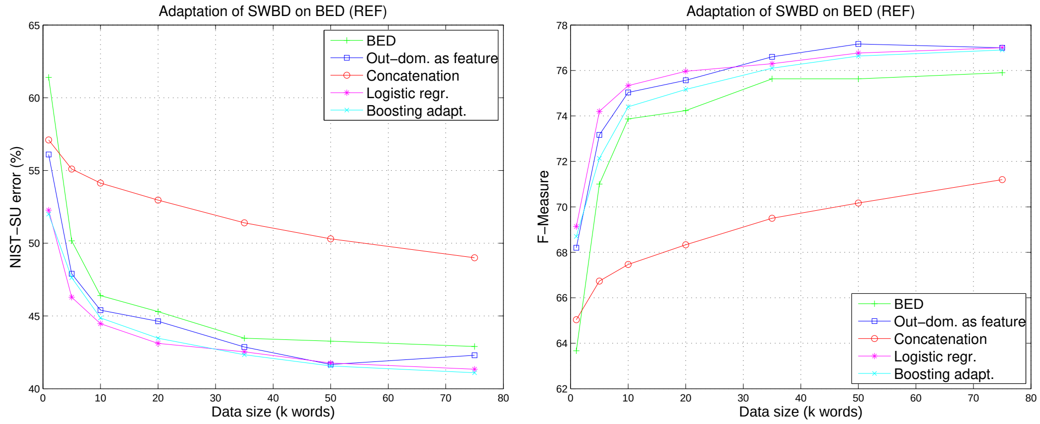


Figure 7.3: Comparison of the logistic and exponential loss functions for the BED meetings of the MRDA corpus. The unit of the data size is the number of words.

7.3.1 SWBD Adaptation on BED

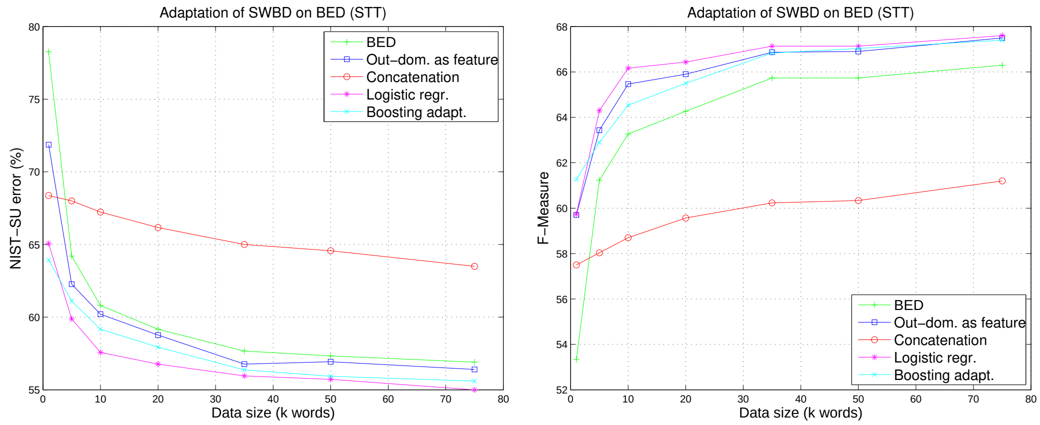
The characteristics of the BED meeting type can be found in Table 6.4 and the ones of SWBD in Table 6.3.

The learning curves in Figure 7.4 and Figure 7.5 show the evolution of the NIST-SU error rate (a) and the F-Measure (b) in reference and STT conditions when the number of training samples of meetings from the BED corpus is increased. In an attempt to smooth the results and hence prevent them from being dependent from noise in the data, all results are averaged on three experiments with three different subsets of the training data as training set. The default acceptance threshold value of 0.5 is used.



(a) NIST-SU (b) F-Measure

Figure 7.4: NIST-SU error rate (a) and F-Measure (b) for the BED classifier evaluated on the BED test set; in reference conditions.



(a) NIST-SU (b) F-Measure

Figure 7.5: NIST-SU error rate (a) and F-Measure (b) for the BED classifier evaluated on the BED test set; in STT conditions.

The learning curve graphs that are presented in this work contain five curves, which correspond to the performance of the four adaptation methods presented in Chapter 5 and the performance of the classifier trained on the in-domain data only. In the case of Figure 7.4 and Figure 7.5, the in-domain classifier is the curve labeled as 'BED'.

We first examine the reference conditions on Figure 7.4. *Boosting* adaptation and logistic regression are the methods that perform the best when there is very little BED training data available. The "feature" adaptation does not achieve such an accurate classification when there is little BED data available, i.e. it does not exploit the SWBD data as well as *Boosting* adaptation and logistic regression in this particular case. As has been discussed in Section 5.5, the poor performance of the "feature" adaptation is a consequence of its strong binding to the in-domain classification model.

The concatenation method performs poorly in comparison to the other methods, independently from the amount of BED data used. As explained in Section 5.5, the weight given to each model of the data is proportional to the amount of data of the model, which makes it impossible for the combined classifier to be efficient even when lots of in-domain data is available, since the out-of-domain data still weigh in the classifier decision.

On the contrary, logistic regression is in the best perforers independently from the size of the BED data used. It reduces the NIST error rate by 17.5% relative for 1k, 8.4% for 3k and 3.7% for 80k, which are all statistically significant improvements according to a Z-test with 95% confidence range.

On the experiment in STT conditions, the "feature" adaptation method performs even worse than the simple concatenation of data. However, the performance of the "feature" method gets better than the concatenation method as soon as 5k of BED data are available. Note that on the REF conditions, the "feature" method always outperforms the concatenation method.

All figures show that the more meeting data, the lower the absolute error rate and the smaller the difference between the classifier trained on BED only and the ones that use SWBD to create a new adapted model. Such a behavior is expected since adding in-domain data to the training set used by the learning algorithm to build the predictive function $f(x, l)$ (and assuming that these data are consistent and reasonably noiseless) the added data allow the classifier to make the predictive function $f(x, l)$ more accurate. However, it should be noticed that even with the full BED training data, all adaptation methods but the data concatenation perform as well, when not better, as the classifier built only on the BED data.

Getting the same error rate with less labeled data

Up to now, we have compared the performance of the BED classifier for a given amount of in-domain data and compared it to the performance of adaptation methods for the same amount of in-domain data (i.e. looking at the different curves for a fixed point on the x-axis). We have in particular observed that independently from the amount of the BED data available, the adaptation methods always outperform the BED classifier. This way of looking at the adaptation performance is however only one of the benefits of adaptation. Indeed, as explained in Chapter 3, adaptation also allows, by using the out-of-domain data together with an amount a_1 of in-domain data, to reach the same classification accuracy as would have been obtained using an amount a_2 of in-domain data only, with $a_1 < a_2$.

This interpretation of the performance can be seen on Figure 7.4 by fixing a point on the y-axis and looking at the x-values for the different curves. For example, looking on the NIST-SU error rate graph, we observe that one would need to use $1k$ of in-domain data together with SWBD data, using the *Boosting* adaptation or the logistic regression method, to reach the error rate of the BED classifier for $4k$ of in-domain data. The difference becomes bigger when more in-domain data is available: with 15k of in-domain data, the error rate yielded by the *Boosting* adaptation method is equal to the one of the BED classifier trained on 35k of in-domain data. Finally, the best performance achieved by the BED classifier (75k of in-domain training data) is equal to the one topped by the *Boosting* adaptation and the logistic regression methods with only 20k of in-domain training data.

Differences between STT and reference conditions

The performances on the STT conditions globally show the same pattern as the performance on the reference conditions, although they are per se 10%-15% lower. Once again, this absolute error rate difference can be explained by the errors made by the ASR in the word recognition phase. Nonetheless, in both STT and reference conditions, one would need to label four times as much in-domain training data (1k vs. 4k) to reach the same first performance, assuming a classifier needs at least 1k of training data to be trained. Approximately, the same ratio holds for the best performance of the whole BED corpus with 75k of data, reached by logistic regression on both STT and reference conditions with only $15 - 20k$ of BED training data.

Besides the lower absolute performance on STT conditions than on the reference ones, the only major difference between the figures 7.4 and 7.5 is that data concatenation outperforms the "feature" method on the STT conditions, while it was always worse than any other adaptation method on the reference conditions. This difference can be explained by the fact that while the overall performance on the STT conditions is worse than the one obtained on the reference conditions by the BED-STT and BED-REF classifiers respectively, the SWBD model is in both cases trained on the reference conditions, which means that it is lexically accurate. Thus, using massively the SWBD data as done by the concatenation method helps when very little BED data are available. On the other hand, having the same errors on the training set and the test set can help. For example, keeping the words that were wrongly inserted by the ASR during the recognition phase on the training set makes sense since it is likely that the ASR is going to do the same insertions on similar instances of the test set. These similarities shared by the training and the test set are however not able to compete with the higher lexical accuracy provided by SWBD when very little data is available.

Because the experiments in STT conditions have shown to have the same pattern that the ones done in reference conditions, we focus in the next experiments on the reference conditions only and compare them with the results obtained in this experiment.

7.3.2 SWBD Adaptation on MRDA

The results for the reference conditions are presented in Figure 7.6 in the same form as in the previous section. Figure 7.7 shows the same two curves but only for the data size from 1k to 80k, which makes it easier to compare the results of the current experiment with the ones presented in the previous section for the SWBD adaptation on BED. Notice that on Figure 7.6, the *Boosting* adaptation curve is stopped at 135k. This is because the *Boosting* adaptation requires a lot of memory and ran out of memory after the 135k point.[2]

The total available amount of MRDA data is 540k, as shown in figure 7.7. When very little in-domain training data is available, adding in-domain data to the training set has the effect of greatly improving the classification performance. This is especially made clear by the almost vertical slope of all adaptation methods (except concatenation) and the MRDA classifier on the extreme left. However, the more available in-domain data is added to the training set, the less effective the repercussion on the classification performance is. After a certain amount of in-domain

[2]This problem will be solved soon, since a new version of Boostexter compiled for 64 bits machines should be released soon.

data is available for the training set, adding in-domain data seems to not improve significantly the classification performance anymore. For example, the “feature” adaptation has a F-Measure value of 80.0 with 135*k* of in-domain data. With 540*k* of in-domain data, the F-Measure value is 80.7. So in this case, adding more than 300*k* of in-domain labeled data to the training set has led to an improvement of less than 1% of the F-Measure.

The asymptotic behavior of the curves tends to an imaginary limit. Although it is hard to estimate such a limit and there is no proof of its existence, one can reasonably assume its existence, based for example on the knowledge that the data set is noisy. Moreover, this limiting behavior has also been observed in prior work, such as in Tur (2005). The limiting behavior also strengthens the effectiveness and importance of using adaptation. For example, the “feature” method reaches the top performance of the MRDA classifier with less than 300*k* of data, which means 240*k* less labeled data. Naturally, when labeling a new corpus, one cannot compare the performance of the adaptation with the performance of the classifier trained on the full in-domain data, since one’s goal is precisely not to have to label the whole corpus. Thus, there is a risk that the performance of the adaptation is lower than the one that would have been obtained with the whole in-domain classifier. However, one could develop means to estimate the top reachable error based on the evolution of the learning curves with little amount of in-domain data.

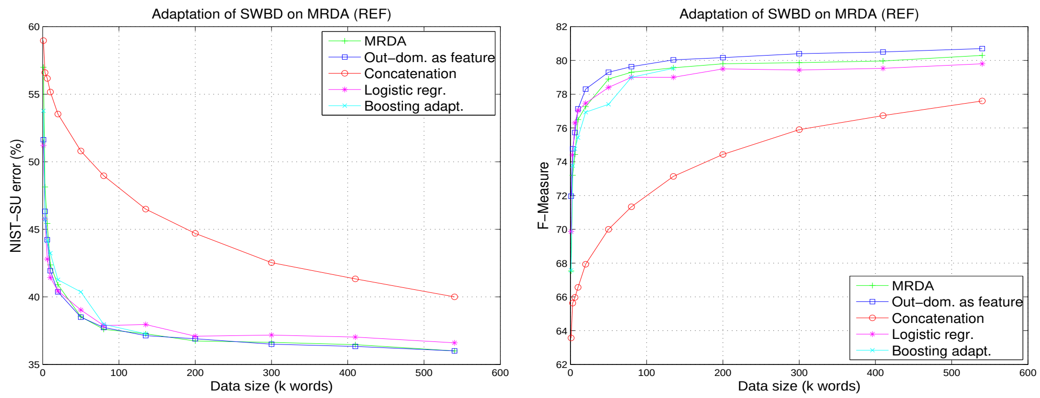
Discussion on the Adaptation Methods

With 50*k* of in-domain training data, we notice that the *Boosting* adaptation curve has a performance that seems out of the natural evolution of the curve. This can be explained by the way the optimization of the weight η, that balances the importance of the out-of-domain model and the in-domain model, is done. Indeed, the optimization η is done by manually trying eight different weights on the development set and choosing the one that minimizes the test error. Although the parameter η used on the test set is averaged over three experiments, there is no warranty that it is the weight which leads to the optimal adaptation between the out-of-domain and the in-domain model. Moreover, and this also holds for both the *Boosting* adaptation and the logistic regression method (and every other methods that would need to optimize some weights on the development set), the data distribution in the development set is different from the one in the test set and an optimal weight for the development set might not be optimal for an evaluation on the test set. These two reasons lead to some suboptimal performance on the evaluation classification.

For the logistic regression method, the optimization of the weights can also be suboptimal in essence. Indeed, as explained in Appendix A, the Newton-Raphson method used to find the optimal weights is an iterative algorithm that may end up finding local optima. This explains the performance of the logistic regression method, which is lower than the one of the MRDA classifier, while it should not since optimal weights should allow the logistic regression method to do at least as well as the MRDA classifier by ignoring the out-of-domain classifier if it hurts. One could argue that the sub-optimality is the consequence of the optimization of the weight on the development, as explained above. However, the uncertainty about the difference between the development set and the test set can be removed by optimizing the weights on the test set. Since the weights are optimized on the same set as the one used for the evaluation of the classifier, the weights are guaranteed to yield the best possible classification performance on this set. Results showed that even with this cheating experiment, the logistic regression method yields lower result than the MRDA classifier.

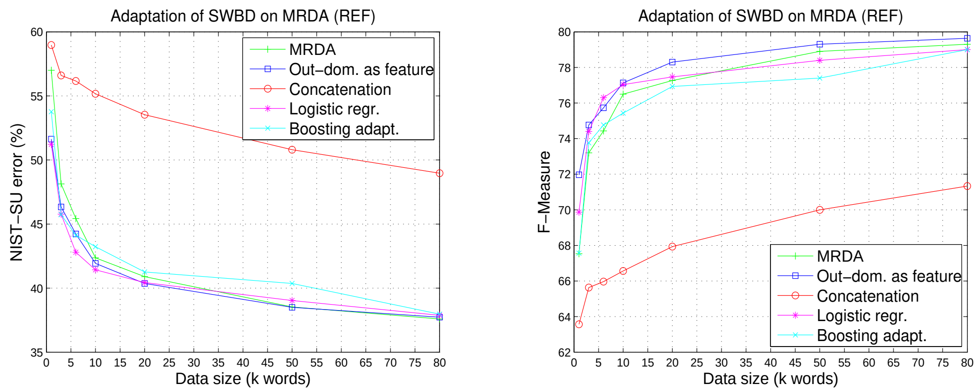
The non linear relation between the prediction and the predictor variables in logistic regression makes it not straight forward to interpret the weights and see if, for instance, the in-domain of the out-of-domain classifier has been given a null importance. The use of linear regression could be useful for this particular case, since the weights in linear regression are easily interpretable.

Although the *Boosting* adaptation and the logistic regression experiment do not perform as well as the MRDA classifier, the best classification performance is still due to an adaptation method,



(a) NIST-SU (b) F-Measure

Figure 7.6: NIST-SU error rate (a) and F-Measure (b) for the adaptation of SWBD on MRDA using the four adaptation methods described in Chapter 5 in reference conditions. The curve named 'MRDA' is the performance obtained by using only the classifier that was trained on the MRDA corpus, i.e. without any adaptation, and can thus be considered as the baseline.



(a) NIST-SU (b) F-Measure

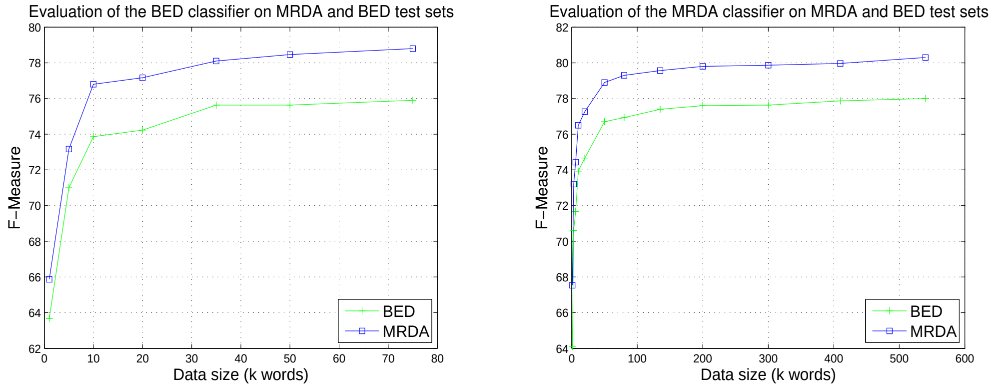
Figure 7.7: Zoom on the first 80k of in-domain training data of Figure 7.6.

i.e. the "feature" method, rather than to the in-domain classifier. Note that this is an inverted ranking compared to the BED experiment of the previous section in which this one method was the worst among the three adaptation methods. Recall that, as explained in Section 5.5, the "feature" method builds a classifier that uses the in-domain data to which the prediction of the out-of-domain classifier has been added as a feature. This is not the case of logistic regression, which uses only the predictions of the in-domain and the out-of-domain classifier, neither is the one of the *Boosting* adaptation which, uses the classifier built on the in-domain data and adapt it using the data of the out-of-domain. The "feature" method is thus, by construction, the one that gives the most importance to the in-domain data. As a consequence, its performance is tightly related to the one of the in-domain classifier. This can be an advantage in the case where the in-domain classifier has a high performance, as in this experiment, but it can also be a drawback if the in-domain classifier performs poorly and the out-of-domain data are able to improve consequently the classification accuracy, as in the previous experiment.

Differences with the BED experiment

Since BED is a subset of the MRDA corpus, this experiment can be seen as an extension of the adaptation of SWBD on BED experiment presented in the previous section. One could therefore expect to observe the same results on both experiments. Some remarks done in the previous paragraph and a comparison of the curves presented in the graphs presented in Figures 7.6 and 7.7 with the ones of 7.4 undermine this hypothesis. One obvious difference can be observed on the performance of all methods with the maximal amount of MRDA data: the MRDA classifier outperforms the logistic regression and concatenation methods and only the "feature" method performs better than the MRDA classifier does (80.7 vs. 80.3 F-Measure). Since the *Boosting* adaptation performance could not be assessed for this amount of data, one could argue that this particular method could have outperformed the MRDA classifier as well. However, looking back at the beginning of the curves, we observe that already for 10*k* and up to 80*k* of in-domain training data, the *Boosting* adaptation method is outperformed by the MRDA classifier. For the last point computed for the *Boosting* adaptation (135*k*), the performance of the *Boosting* adaptation method yet comes back to the level of the MRDA classifier (79.5 F-Measure for both of them).

Even though *Boosting* adaptation is the weakest of the adaptation methods (apart from concatenation), the fact that it is outperformed by the MRDA classifier is not due to a weakness of this particular adaptation method. Indeed, the logistic regression method has a lower F-Measure than the MRDA classifier for every evaluation done with 50*k* and more in-domain training data. Recall that on the BED experiment with the exact same SWBD classifier, *Boosting* adaptation and logistic regression both outperformed the in-domain classifier independently of the amount of in-domain data. The BED data being 1/5 of the MRDA data and the other 4/5 being supposedly similar to the BED data makes this difference between the experiment on BED and MRDA difficult to explain. Two reasons can however be propound to explain this difference. First, the MRDA test set seems to be easier to classify than the BED test set for the MRDA and BED classifier as shown in Figure 7.8. The two graphs on this figure show the performance of the BED and the MRDA classifiers respectively, evaluated on both the MRDA and the BED test set. In both cases, the performances of the classifier are significantly better when evaluated on the MRDA test set. While the two presented graphs tend to show that the MRDA test set is easier to classify than the BED test, the performance of the SWBD classifier contradicts this assertion. Indeed, the NIST-SU error rate of the SWBD classifier of the BED test set is 60.4% while the NIST-SU error rate of the same SWBD classifier on the MRDA test set is 61.3%. Thus, the SWBD corpus fits the BED subset of MRDA better than the whole MRDA corpus. The more accurate performance of SWBD on BED is confirmed by looking at the concatenation curves on both graphs: with 1*k* of in-domain training data, the NIST-SU error rate on BED is 1.9% absolute lower than the one of concatenation on MRDA (57.1% vs. 59.0%). Adding in-domain data reduces the difference between the two error rates, such that for 75*k* of data they are identical (49.0).



(a) BED (b) MRDA

Figure 7.8: Evaluation of the BED (a) and the MRDA (b) classifiers on the BED test set and the MRDA test set. The legend indicates the test set.

The differences between the evaluations on the BED and the MRDA test sets mentioned in the previous paragraph are partly due to a sampling accident, since an evaluation on the development set shows a lower difference in the classification performances. However, even though these differences are less obvious on the development set than on the test set, they are still repeated on the development set and incite to think of more genuine differences between the BED corpus and the rest of the MRDA corpus. In the next section, we examine in more detail the differences between the meeting types of the MRDA corpus by means of lexical statistics and another adaptation experiment.

7.4. Adaptation within the MRDA Meeting Types

As already mentioned above, the MRDA corpus is divided into three main types of meetings, BED, BMR and BRO. While these meetings have all been recorded using the same settings, their speakers as well as the subjects that they discuss are yet different, as has been explained in more detail in Section 6.1. Therefore, proceeding to some kind of adaptation on them is not pointless and can even be useful in practice, although they technically all belong to the same corpus. Imagine for example that one decides to extend the MRDA corpus with a new type of meeting that would have been recorded with the same settings as the three already existing meeting types. The existing meetings could be used to help label the new recorded meetings.

Concretely, one could train a classifier on the existing meetings and use it to output predictions for the new meetings, assigning a confidence to each of the prediction output. The instances with low confidence could then be given to human labelers, who would thus get a much smaller amount of data to label. This procedure is called *active learning* and has been studied by Tur (2005) for the case of spoken dialog system. Another approach could be to use the study on the three already labeled types of meetings to estimate the amount of data of the new meeting type that would need to be labeled in order to reach an error rate under a certain percentage. The labelers could consequently estimate the amount X of data that the use of the already existing labeled data could save them from labeling.

More experiments concerning the meeting types within the MRDA corpus have been done and the interested reader can found them in Section B.1.

	BED test	BED		BRO		BMR	
# words	2486	4257	665	4731	830	6681	589
# bigrams	16036	34563	9445	48562	9497	66846	8373
# trigrams	27221	67600	22723	108885	22418	148590	21523

Table 7.3: Lexical statistics for the test set of BED (first column) and the training sets of the three meeting types of MRDA. For each training set the first column shows the absolute number of distinct words (resp. bigrams and trigrams) that appear in the training data. The second column shows the number of words (resp. bigrams and trigrams) that appear in the BED test set and do not appear in the training set of the meeting type.

7.4.1 BRO Adaptation on BED

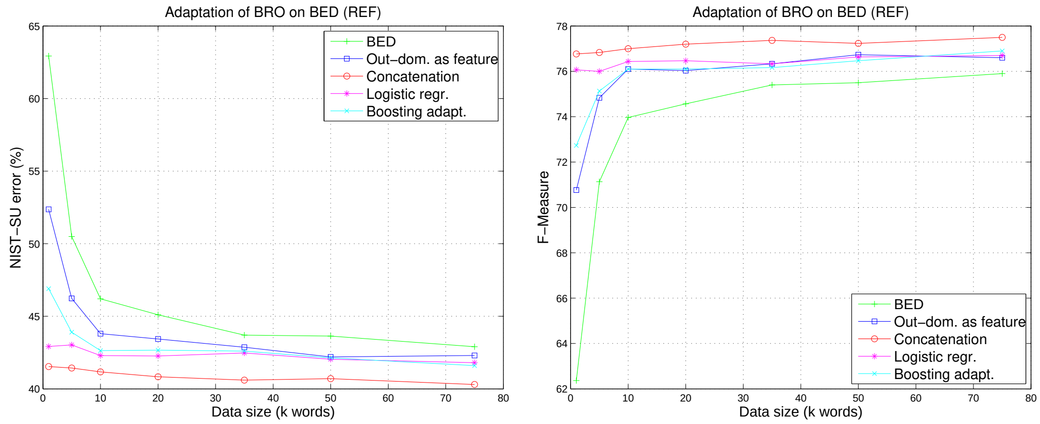
In this section, we perform the same kind of experiment as in Subsection 7.3.1 and Subsection 7.3.2. As observed in Section B.1, the BRO classifier performs better on the BED and MRDA test sets than the BMR classifier when compared with an identical amount of in-domain training data. We therefore chose the BRO corpus as the out-of-domain. The results of these experiments are reported in Figure 7.9 as before in a shape of learning curves for both the NIST-SU error rate and the F-Measure.

As for the adaptation of SWBD on BED, *Boosting* adaptation, logistic regression and "feature" adaptation methods all perform better than the BED classifier independently of the amount of in-domain training data available. The major difference with the SWBD on BED experiment is the performance of data concatenation which is significantly better than all other methods: 2% absolute better for every amount of in-domain training data size. This outstanding performance of the concatenation method is even more surprising when compared with the SWBD adaptation on BED and the SWBD adaptation on MRDA, where the concatenation was each time the worst method. The difference between these two previous experiments and the current one is that in the current one, both the in-domain and the out-of-domain come from the same corpus. Therefore, even if they have some lexical and structural differences as explained above, the distribution of their training data are still very near from each other. Thus, one can think that concatenating data from the out-of-domain is in this case the same as adding data from the in-domain.

The other adaptation methods show once again the same patterns as observed in the previous experiments: logistic regression and *Boosting* adaptation are very near from one another, with logistic regression performing better with little in-domain data. The "feature" method is the one that sticks the most to the in-domain classifier performance, that is to say in this case the one that performs the worst. They all converge to the same performance when the amount of in-domain data is increased.

7.5. The Importance of Features

The learning task consists of some data that have to be classified into classes on one side, and a learning algorithm able to classify a set of data on the other side. The question is: how can the data and the learning algorithm interact? In other words, how can one represent a set of data such that a learning algorithm can handle them. Features play the intermediary role between the learning algorithm and the data by creating a representation of the data. The choice of the features is crucial since they contain all the knowledge that the learning algorithm has about the data. So the challenge is to make the features represent the data in a way that discriminates the classes, so that the classifier built by the learning algorithm has a higher probability to put each instance of the data in the correct class.



(a) NIST-SU

(b) F-Measure

Figure 7.9: NIST-SU error rate (a) and F-Measure (b) for the BED classifier by itself and the same BED classifier combined with BRO using the same four adaptation methods as before.

7.5.1 Motivating Example

In order to better understand the importance of features, let us consider the problem of classifying cars into three exclusive categories: sport cars, SUVs and standard cars. What features would one choose to discriminate these three types of cars? Intuitively, a low engine power would most likely mean that the car is a standard car, and a high engine power that the car belongs to the SUV or sport car categories. To distinguish those two categories, the height of the car could be a good choice, since SUVs are usually bigger than sport cars. So by using the engine power and the height of one car, it intuitively seems that one can classify cars into the three categories standard cars, SUVs and sport cars. In other words, given the height and the engine power of a car, a classifier should be able to tell if the car belongs to the category sport cars, standard cars or SUVs.

However, let us imagine that instead of the height and the engine power of a car, one had chosen the year of construction of a car and the country in which it was built as features. The year of construction and the origin country are two features that are not helpful at all to distinguish the three categories of cars considered in this example. For example, knowing that a car was built in 2001 in Germany does not tell anything about its belonging to either of the three classes. The year of construction and the origin country are not discriminative for the classification task desired in this example. It does not mean that these two features are wrong or do not describe properly the cars, it only means that they are not appropriate for this particular task.

Coming back to our primary choice of features, we have two features, the height and the engine power, that intuitively would help to perform the classification into the three classes SUVs, standard cars and sport cars. Now, these features will probably not be enough to prevent the learning algorithm from misclassifying some cars. For example, it might be that a car considered as a standard car be powerful and thus ends up being classified as a sport car. So we will need to add features to describe a car more precisely to the learning algorithm. But what features? Could the length of the car help? What about the color?

In most cases it is impossible to know in advance what features will help or hurt for the given classification task, for at least two reasons. The first reason is that it is not straight forward to know if a feature distribution will be correlated with the classes distribution. For example, would the color of a car help classify a car in one of the three categories? It is difficult to say, since the common sense would say that sports car are generally more flashy than standard cars, but some standard cars also have flashy colors. The second reason that makes it difficult to know if an additional feature will help to predict more accurately the class of an instance (a car in the current example), is that even if the distribution of the new feature is correlated with the distribution of the classes, it might also be correlated with the distribution of a feature which is already in the feature set, and hence not bring any improvement. In the case of the cars example, the height and the weight of a car are two features that might be strongly correlated (although counter-examples can always be found). In the case of sentence segmentation, Eisenstein & Davis (2005) have shown that although gestural features are correlated with sentence segmentation, adding them to classification models that already contain prosodic and lexical features does not make the classification more accurate because the gestural features are correlated with the prosodic and the lexical features.

7.5.2 Subset of Features

In Section 6.3, we presented four different subsets of features to describe the data: one containing lexical features only (designated as the *lexical* set in the rest of the section), one with prosodic features only (*prosodic*), the third one with lexical features and the pause duration between two words (*lexical+*) and finally one containing all available features, that is all prosodic and lexical features (*all*).

In this section, we study and compare the performances of the classification models built on each of these four subsets of features. We use the MRDA and TDT4 corpora, since the prosodic features for the SWBD corpus were not available. For the MRDA corpus, we had the prosody information only for the reference conditions. For every measure reported in the tables, the acceptance threshold has been optimized on the development set. The optimization was done by trying all thresholds between 0.1 and 0.9 with a step of 0.1 between each of them. The threshold that yielded the best NIST-SU error rate (respectively F-Measure) on the development set was chosen and used as the acceptance threshold on the test set.

The results (F-Measure and NIST-SU error rate) are presented in Tables 7.4, 7.5, 7.6 and 7.7 for the reference (7.4, 7.5) and the STT conditions (7.6, 7.7). The purpose of this section is to compare the performance of one classifier through the four different sets of features – this is done by reading the tables column-wise. For example in Table 7.4, the first column shows the performance by means of the NIST-SU error rate of the MRDA classifier evaluated on the MRDA test set for the four sets of features. We observe that the best performance in this case is topped by the classifier that used the *all* features subset, followed by the *lexical+* subset, than the *lexical* set and finally the *prosodic* features set. Considering that the *all* features set uses 43 features, the performance of the same classifier with only the 9 features of the *lexical+* features set is remarkably high, since the *all* classifier uses 34 more features than the *lexical+* one (36.5% vs. 35.6% NIST-SU error rate). One could be tempted to conclude from this observation that all the prosodic features added in the *all* features set are not that helpful.

Pause Duration: the most discriminative feature

Concluding that the prosodic features are not helpful is however wrong. First, and even if the absolute error rate gain is little (0.9%), the relative error gain is 2.5%, which is statistically significant. Second, the performance of the classifier with only these prosodic features is only 4.9% absolute worse than the one yielded by the classifier that was fed with only lexical features (53.7% vs 48.8%). Since the NIST-SU error rate of a classifier that hypothesizes sentence boundaries everywhere (also called *chance* performance) is by definition 100%, this performance shows that prosodic features are indeed useful for the detection of sentence boundaries. The little change observed when adding them to the *lexical+* feature set is partly due to the presence of the *pause duration* feature in the *lexical+* feature set.

Classifier:	MRDA		TDT4	
Test set:	MRDA	TDT4	TDT4	MRDA
Lexical	48.8	100.3	93.4	72.5
Prosodic	53.7	92.2	65.5	73.3
Lexical+	36.5	97.8	56.4	65.3
All	35.6	93.3	49.2	60.4

Table 7.4: NIST-SU error rates for the classifiers built on the MRDA and TDT4 corpora for the four different sets of features. The two classifiers are evaluated on both the MRDA and the TDT4 test sets for each set of features. In reference conditions.

Classifier:	MRDA		TDT4	
Test set:	MRDA	TDT4	TDT4	MRDA
Lexical	74.2	35.5	48.0	60.5
Prosodic	69.9	52.9	67.4	64.7
Lexical+	80.9	55.9	72.5	66.2
All	81.7	58.3	75.5	68.3

Table 7.5: F-Measure results for the classifiers built on the MRDA and TDT4 corpora for the four different sets of features. The two classifiers are evaluated on both the MRDA and the TDT4 test sets for each set of features. In reference conditions.

Classifier:	MRDA		TDT4	
Test set:	MRDA	TDT4	TDT4	MRDA
Lexical	70.9	101.9	97.7	86.6
Prosodic	NA	NA	78.1	NA
Lexical+	49.5	99.3	76.2	72.4
All	NA	NA	71.2	NA

Table 7.6: NIST-SU error rates for the classifiers built on the MRDA and TDT4 corpora for the four different sets of features. The two classifiers are evaluated on both the MRDA and the TDT4 test sets for each set of features. In STT conditions.

Classifier:	MRDA		TDT4	
Test set:	MRDA	TDT4	TDT4	MRDA
Lexical	62.4	29.0	39.7	51.4
Prosodic	NA	NA	58.0	NA
Lexical+	73.3	47.2	59.5	62.8
All	NA	NA	61.8	NA

Table 7.7: F-Measure results for the classifiers built on the MRDA and TDT4 corpora for the four different sets of features. The two classifiers are evaluated on both the MRDA and the TDT4 test sets for each set of features. In STT conditions.

Indeed, the pause duration between two words is intuitively a very natural indicator of the presence or absence of a sentence boundary. This intuition is confirmed by the 12.3% absolute NIST-SU error rate gained between the *lexical* features and the *lexical+* features set, which is the same set with only the pause duration added to it. Another indicator of the importance of the pause duration feature can be seen in the structure of the *Boosting* classifier. Recall that at each of the 1000 iterations of the learning process, *Boosting* builds the new basic rule that helps to reduce the most the current classification error rate. This rule is based on one feature, and counting the number of times that one feature is used to build a basic rule along the learning process gives us an indication of its importance. The pause duration feature appears 82 times out of the 1000 iterations in the MRDA classifier built on the *all* feature set; it is from far the most used feature, followed by the windows energy difference, the words energy difference and the pitch windows difference measures.

Redundancy of the prosodic and the lexical models

The second reason that can be put forward to explain the little performance difference between the *lexical+* and the *all* sets of features is the *redundancy* of the information conveyed by the prosodic model and the lexical model. The information contained in the two models is not exactly the same, since as pointed out before, merging the two sets of features allows us to reduce the error rate of 2.5% relative. Even though it is difficult to prove formally this redundancy between the lexical and the prosodic models, it could be an explanation for the use of all lexical features not improving drastically the performance towards the *lexical+* feature set. Moreover, the prosodic features amongst themselves also have a huge degree of redundancy, since among the 43 features, most of them are only different measures of the same physical phenomenon. For example, the energy differences between to consecutive words w_1 and w_2 generates four features: the difference between the maximum energy of w_1 $E_{w_1}^{max}$ and the minimum energy of w_2 $E_{w_2}^{min}$, the difference between the two maximums $E_{w_1}^{max}$ and $E_{w_2}^{max}$, the difference between the two minimums $E_{w_1}^{min}$ $E_{w_2}^{min}$ and the difference between $E_{w_1}^{min}$ and $E_{w_2}^{max}$.

Differences between the MRDA and TDT4 corpora

Back to Table 7.4 and Table 7.5, one observes that the ranking that had been established for the MRDA classifier evaluated on the MRDA corpus also holds for the TDT4 classifier evaluated on the MRDA test set: first the *all* features set, then the *lexical+* one, the *lexical* one and finally the *prosodic* one.

Now if we look in more detail at the first and the third columns, we observe some slight differences between the MRDA and the TDT4 corpora. First, the ranking among the feature sets for the classifier built on the TDT4 corpus and evaluated on itself is not the same as was described above for the MRDA corpus: the experiment with the *lexical* features set performs much worse than the one with the *prosodic* features set (65.5% vs. 93.4% NIST-SU error rate and 67.4 vs. 48.0 F-Measure).

This result suggests that either lexical features have less accuracy on the TDT4 corpus than they have on the MRDA corpus, which the high NIST-SU error rate (93.4%) on the lexical experiment tends to confirm, or that prosodic features are more accurate on the TDT4 corpus than on the MRDA corpus. Since the TDT4 corpus contains more structured speech, one could indeed expect the prosodic information to be more reliable, all the more because the broadcast news speakers are professional speakers and should therefore particularly stress the intonation in their sentences as well as the pauses between their sentences. The effects of prosodic features will be discussed in more detail below.

On the contrary, the well-structured speech of TDT4 seems to weaken the lexical information, at least for the sentence segmentation task. This is confirmed by Table 7.5.2, which shows statistics on the words ending a sentence in both MRDA and TDT4 training data sets. As noticed in Section 6.1 on the description of the data, the MRDA corpus has a higher absolute number of sentence units than TDT4, although the number of words in TDT4 is higher. However, the number of different words that end a sentence units is much smaller on MRDA than TDT4. Thus a word in MRDA that ends at least one sentence unit in MRDA ends on average 14.7 sentence units, while this number on TDT4 is only 7.4. This already gives a clue of why the lexical model is stronger on MRDA than on TDT4.

Tables 7.9 and 7.10 show the percentage of sentence units ended (resp. started) by the five words that end (resp. start) the more sentence units in each corpora. The word that ends the most sentence units in MRDA ("`yeah`") ends more than 10% of them, while the most frequent one in TDT4 is the `@reject@` token of the ASR and appears 1.85% of the cases. The `@reject@` token of the ASR (which is always followed by a sentence boundary) being the most frequent word to end a sentence indicates that there is no strong lexical pattern for the sentence unit ending mark in TDT4. On the contrary, the words that appear the most frequently on the end of a sentence unit express that the MRDA data have a lexical repetitive pattern that is proper to the conversational speech style.

The difference between the TDT4 and the MRDA corpora is weaker on the words starting a sentence unit. The presences of "`and`", "`i`" and "`the`" are expected among the five most frequent words that start a sentence. The presence of the hesitation word "`uh`" is more surprising for TDT4, since the speakers on broadcast news, being professional speakers, are expected to be fluent.

On the meetings, the word "`yeah`" is the most frequent one to start a sentence boundary. Recall that it was already the most frequent to end a sentence boundary. This is explained by the double usage of the word "`yeah`": while "`yeah`" can be used to grab the floor and confirm what someone has just said, or answer a question, it is also frequently used in meetings to acknowledge in the background what the foreground speaker says (backchannel). Thus, 2% of the sentences in the MRDA corpus begin with the word "`yeah`" and 39% of the backchannels in the same corpus are a simple "`yeah`". Furthermore, the five most frequent words to start a sentence unit are words that can be used as backchannels.

	MRDA	**TDT4**
Number of SUs	82,435	77,262
Number of distinct words that end an SU	5,595 (50.7%)	10,453 (32.5%)
Number of distinct words that start an SU	1,955 (17.7%)	4,733 (14.7%)
Avg. nb. of SUs for each word ending an SU	14.7	7.4
Avg. nb. of SUs for each word starting an SU	42.1	16.32

Table 7.8: Statistics on the words ending a sentence unit (SU) on the MRDA and the TDT4 corpora. In the second and third rows, the percentage indicates the percentage of words in the vocabulary of the corpus that start (resp. end) and SU. The vocabulary size were presented in Table 6.3

Prosodic features

We mentioned above that the high reliability of the prosodic features in the TDT4 corpus can be the cause of the performance gap between the *prosodic* experiment and the *lexical* experiment by the TDT4 classifier on the TDT4 test set. This gap is all the more remarkable because the performance of the *prosodic* and *lexical* experiment on the MRDA corpus are inverted, i.e. the lexical features achieve a better performance than the prosodic ones. When using the TDT4-REF

	MRDA		TDT4	
word 1	yeah	(10.82%)	`@reject@`	(1.85%)
word 2	uhhuh	(4.32%)	it	(1.16%)
word 3	right	(3.96%)	today	(0.72%)
word 4	okay	(2.91%)	that	(0.65%)
word 5	um	(2.48%)	yesterday	(0.62%)

Table 7.9: The five words that end the more sentence units in each corpus and the corresponding percentage of sentence units that they end.

	MRDA		TDT4	
word 1	yeah	(11.20%)	`@reject@`	(11.11%)
word 2	and	(9.42%)	the	(8.51%)
word 3	so	(8.09%)	and	(4.28%)
word 4	i	(6.31%)	uh	(3.63%)
word 5	but	(5.14%)	i	(3.24%)

Table 7.10: The five words that begin the more sentence units in each corpus and the corresponding percentage of sentence units that they begin.

classifier, the performance on the MRDA test set follows the same order as on the TDT4 test set for the F-Measure: the prosodic features outperform the lexical ones. In fact the performance yielded by the experiment on the *prosodic* feature set show that the variability of the prosodic features does not depend as much on the corpus as the variability of the lexical features. As an illustration, the F-Measure score is higher than 50 in all cases and if we exclude the result of the MRDA classifier on TDT4, all F-Measure scores are around 65 to 70. This is clearly not the case of the *lexical* experiment, whose performances are much more variable depending on the training and testing corpora (F-Measure going from 35.5 to 74.2).

The little variability of the prosodic features make them ideal candidates for adaptation from the TDT4 to the MRDA corpus and vice versa. In particular, this could lead to the idea of using an out-of-domain classification model without any lexical information and to adapt it to the in-domain by using a lexical classification model of the in-domain.

STT conditions

While the above-mentioned observations about the results are mainly based on the results in reference conditions, we also report the same results in STT conditions in Tables 7.6 and 7.7. These results are however sparse, since the prosodic features for the MRDA corpus in STT conditions were not available. As already mentioned in the preceding experiments comparing the STT and reference conditions, we again observe that the general pattern (classifier ranking, feature subset ranking) is the same as on reference conditions, but the absolute performance measured by the NIST-SU error rate and F-Measure is lower. An example of this can be noticed by comparing Table 7.4 and Table 7.6 row by row. For instance in the first row, although the gap gets smaller when the errors are big, the results in reference conditions outperform the ones in STT conditions in every case.

CHAPTER

8

Conclusions and Outlook

Sentence segmentation from speech is a difficult task because of the high variability of speech. The high variability of speech can especially be noticed in the differences of the speech characteristics among various speech styles. In an effort to reduce the labeling of data and to improve the accuracy of the sentence segmentation on a given in-domain, we aimed at using data from an out-of-domain with a potentially different speech style from the in-domain.

Throughout this work, we studied three speech styles: broadcast news, telephone conversation and meeting speech style. We first showed that the sentence segmentation performance of a classifier trained on one speech style is not accurate when applied to a speech style with different characteristics, such as the average sentence length and the number of speakers.

The low sentence segmentation accuracy obtained when segmenting data from the in-domain data with a classifier built on the out-of-domain data justified the need for adapting the classifier built on the out-of-domain to the in-domain. We presented various model adaptation methods for the sentence segmentation task and experimentally assessed them.

In particular, we showed that using a model adapted from telephone conversations to perform the sentence segmentation task on meeting data is useful in two ways. First, the adapted model achieves the same classification accuracy as the in-domain model with less in-domain data, i.e. the adapted model *reduces the amount of labeled meetings data needed* to achieve a given performance. Second, the adapted model was also shown to *improve the accuracy* of the classifier built on the entire meeting data set.

Among the various methods presented, the optimal adaptation method was shown to be dependent on the amount of in-domain data available and on the level of classification difficulty of the in-domain data. When very little in-domain data is available, the methods that can exploit the best the out-of-domain classifier, such as logistic regression and boosting adaptation, performed the best; on the contrary, when a large amount of in-domain data is available, the methods that allow the adapted model to stick to the in-domain model and give less importance to the out-of-domain, such as the "feature" method, were the most efficient. All methods that we presented, although used only for the sentence segmentation in this work, could be used for other tasks.

The variability of speech between the speech styles gives a particular importance to the features, since the features that are the most discriminative on one domain are not necessary the most discriminative the other domain. The study of the lexical and prosodic features used in this work confirmed the importance of the pause duration. A comparison of the lexical and the prosodic features showed that the features that allow the most accurate performance depend on the speech style.

In particular, results showed that the lexical features were stronger indicators of sentence boundaries than prosodic features in meetings, whereas it was the contrary for the broadcast news. On both speech styles the best accuracy of the sentence segmentation task was performed using the prosodic and the lexical features. Finally, we showed that the accuracy of a classifier on the out-of-domain was nearer the performance of the in-domain classifier when using prosodic features rather than when using lexical features. We concluded that, at least on broadcast news and on meetings, the prosodic features are more independent from the speech style than the lexical features.

Future work

This work presented model adaptation methods for the sentence segmentation task. We studied model adaptation in both STT and reference conditions, using three speech styles: meetings, broadcast news and telephone conversations. We also examined the differences between these three speech styles by comparing their performance with various subsets of features.

As a continuation of the current study and to show the effect of the model adaptation on further tasks, one could study the effect of the sentence segmentation performed by an adapted model on the dialog act segmentation and classification tasks.

Another challenge is to further improve the adaptation methods presented in this study. In particular, one could aim at finding a method that performs as well as *Boosting* adaptation and logistic interpolation when only a small amount of data is available, and as well as the "feature" method when a large amount of in-domain data is available. This could either be done by finding a new method or interpolating the presented methods.

The improvement of the adaptation is closely related to the features used by the classifier. Finding features that are as independent from the corpus as possible is particularly promising. A general classification model could be built on these adaptable features from an out-of-domain corpus and adapted to the in-domain, i.e. adaptation would not be used as a combination of two domains on the same level but as a specialization of one general model to a more specific one.

In an effort to reduce the labeling work, *active learning* should definitely be considered. The goal would be to automatically select and label only a subset of the instances of the in-domain, depending on the knowledge provided by the out-of-domain. The promising field of labeling data through games is another challenge that could be tried for the sentence segmentation labeling.

APPENDIX

A

Logistic Regression

Logistic regression is part of the generalized linear models category. It allows to make a discrete prediction from a set of variables that can be continuous or discrete, called the predictor variables. The output is usually dichotomous (or binary) and indicates the presence or the absence of a feature. In the case of sentence segmentation task, the output variable indicates the presence or the absence of a sentence boundary. Although the output is binary, the relation between the predictor variables and the response variable is the following real-value function:

$$p(x) = \frac{e^{(\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_t x_t)}}{1+e^{(\alpha+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_t x_t)}}, \tag{A.1}$$

where β_0 is a constant and β_i is the coefficient of the i^{th} predictor variable. The binary output is found from the response variable. The parameter α adjusts the probability p for the cases where x_i are zero, while the parameters β_i adjusts how quickly the probability p changes when the x_i change. Note that these β_i do not have a straight forward interpretation, as is the case in linear regression for example, since the relation between p and x_i is non linear. Multiplying both the numerator and the denominator with $e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_i x_i)}$, we can rewrite Equation A.1 as follows:

$$p(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_t x_t)}} \tag{A.2}$$

Equation A.1 can also be written in the linear way:

$$p'(x) = ln\left[\frac{p(x)}{1-p(x)}\right] \tag{A.3}$$

$$= \beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_t x_t \tag{A.4}$$

by assuming the following transformation:

$$p'(x) = \ln\left(\frac{p(x)}{1-p(x)}\right)$$

The goal is to minimize the error done by approximating p. While some models do have a mathematical solution that minimizes the sum square residuals, there is no such mathematical solution for the logistic curve. The alternative solution is to use a maximum likelihood loss function, where we estimate the parameters of the model such that the computed likelihood (or conditional probability) minimizes the error. The estimation of model parameters is done by an iterative procedure called Newton-Raphson method.

We show the formalism for the binary classification case, since it is the one we use in this work. We denote the set of classes with:

$$\mathcal{C} = c_0, c_1$$

and the set of parameters of the model by:

$$\theta = \{\beta_0, \beta_1, \ldots, \beta_i\}.$$

Moreover, we denote $y_i = 0$ if a sample $x_i \in c_0$ and $y_i = 1$ if $x_i \in c_1$. We want to maximize the conditional probability for an example x_i of being associated with its class c, given the parameters θ of the model:

$$p_c(x_i; \theta) = Pr(C = c \mid X = x_i; \theta)$$

We are given a set of training data with N samples that are pair matching a data example x to a class of $c \in \mathcal{C}$. Since all samples are assumed to be independent from each other, the posterior probability for the N samples can be written as the product of all single probability:

$$\prod_{i=1}^{N} p_{c_i}(x_i; \theta)$$

The log-likelihood of this expression is therefore:

$$l(\theta) = \sum_{i=1}^{N} \log p_{c_i}(x_i; \theta) \tag{A.5}$$

In the case of the binary classification, since either $y_i = 0$ or $1 - y_i = 0$, we have:

$$\log p_{c_i}(x; \theta) = y_i \log p(x; \theta) + (1 - y_i) \log(1 - p(x; \theta)),$$

with θ being the vector of parameters $(\theta_0, \ldots, \theta_t)$. Therefore the conditional log-likelihood of Equation A.5 can now be rewritten as:

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{N} \log p_{c_i}(x_i; \theta) && \text{(A.6)} \\ &= \sum_{i=1}^{N} [\log p_{c_i}(x; \theta = y_i \log p(x; \theta) + (1 - y_i) \log(1 - p(x; \theta))] && \text{(A.7)} \end{aligned}$$

Going back to Equation A.2, we see that we can rewrite $p(x; \theta)$ and $1 - p(x; \theta)$ as:

$$\begin{aligned} p(x; \theta) &= \frac{e^{\theta x}}{1 + e^{\theta x}} && \text{(A.8)} \\ 1 - p(x; \theta) &= \frac{1}{1 + e^{\theta x}}, && \text{(A.9)} \end{aligned}$$

which once substituted in Equation A.7 gives:

$$l(\theta) = \sum_{i=1}^{N} \left[y_i x_i - \log(1 + e^{\theta x_i}) \right]. \tag{A.10}$$

Equation A.10 is the expression that we want to maximize. To find its maximum, we need to find the first order derivative $\frac{\partial l(\theta)}{\theta}$ and set it to zero. This yields a set of nonlinear equations which we can solve by using the Newton-Raphson algorithm. The mathematical developments needed for the Newton-Raphson algorithm are beyond the scope of this work and we refer the reader to Kolen & Brennan (2004); Neter & Wasserman (1974) for more details.

The Newton-Raphson algorithm is an iterative method. At the beginning, every β_t in θ are set to 0. The estimated class y_i of each example $x_i \in N$ are then initialized according to the class they belong to. The iterative procedure is then started: first it computes the probability $p(x_i;\theta)$. Next, using special matrices and vectors of the Newton-Raphson algorithm and the probability $p(x_i,\theta)$ that has just been estimated, a new θ is computed that reduces the classification error. These steps are repeated until a stopping criterion is met.

APPENDIX

B Additional Experiments on MRDA Meeting Subtypes

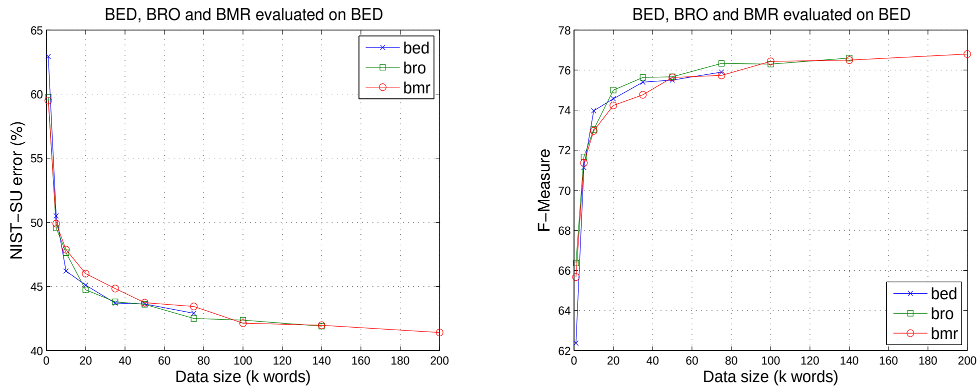
In the following section, we study in more detail the differences between the meeting subtypes in the MRDA corpus (BED, BMR and BRO). We train a classifier on each of the three corpora and evaluate them on the BED test set. The results of this experiment are compared to the ones of a chance experiment with the aim of discovering if the lexical and environmental differences between the three corpora, shown in Chapter 6, influence the classification model built on these corpora.

B.1. Evaluation of BED, BRO and BMR on BED

The BED meeting type was chosen to perform the adaptation, since it is the one which contains the less data and it is more realistic that in practice the in-domain has less data than the out-of-domain. Figure B.1 shows the performance of the three classifiers trained on the BED, BRO and BMR data, and evaluated on the BED test set.

First, notice that note all curves have the same number of points, because the amount of data in the corpora BED, BRO and BMR is different: BMR contains approximatively 200*k* of training data, versus 145*k* and 80*k* for BRO and BED respectively. At first glance, all classifiers seem to have the same performance, with little variations of maximum 2% absolute NIST-SU error rate. With only 1*k* of in-domain data, the performance of the BED classifier is the worst of the three. This is unexpected, since the test set was built on the same corpus – but not the same set – as the BED classifier and a better performance of the BED classifier could thus be expected. Indeed, when two data sets D_1 and D_2 come from the same source, a basic intuition expects the classifier built on D_1 to have a higher accuracy on D_2 than a classifier built on a set of data that do not come from the same source.

The reasons why the BED classifier performs less accurately on the BED test set than the other classifier are not obvious. It was observed in the previous section that the BED corpus was more difficult to classify than the MRDA corpus. This is shown again by the better performance obtained by the same classifier on the MRDA test set in Figure B.3. This repeatedly proven higher difficulty in the classification of the BED corpus, even by a classifier built on BED itself, could find an explanation in some noise in the BED data. The noise – whose origin should be examined – together with the relatively small amount of BED data, would make the BED meeting type more difficult to classify and thus also turn a classifier that learned rules on these data less accurate.



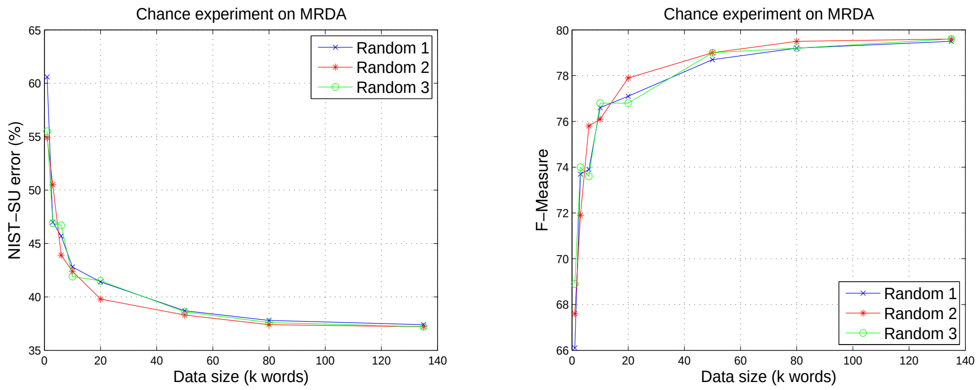
(a) NIST-SU (b) F-Measure

Figure B.1: NIST-SU error rate (a) and F-Measure (b) for the three meeting types BED, BMR and BRO evaluated on BED.

More precise measures on the BED data should however be taken before one can definitely qualify the BED data as more noisy than the BRO or BMR data. Although the reported results have been averaged on three experiments which had different test sets, there is still a chance that the poor BED performance is the consequence of some randomness. The crisscross of the curves on both graphs of Figure B.1 supports this explanation, since adding more data seems to make the performance differences of the three classifiers become narrower. For example, for 50k of in-domain data all NIST-SU error rates are the same; with 75k, the maximal difference is between the three NIST-SU error rate is less than 1.0% absolute (0.9% between BRO and BMR).

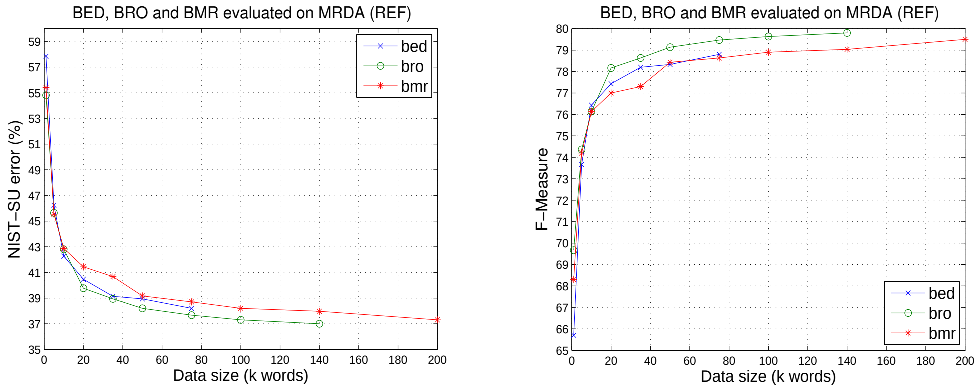
To prove that the results shown in Figure B.1 are rather caused by some randomness in the selection on the data than by the differences among the meeting types, a further "chance" experiment was tried. Three new sets of data are built from scratch independently from their original meeting type. On the same model as in the experiment that involved BED, BMR and BRO, one classifier is built on each of these new data sets and evaluated on the MRDA corpus. The purpose of such an experiment is to see whether the same pattern as on Figure B.1 appears, in which case the behavior of BED, BMR and BRO would be shown to be similar to randomness. The results of the "chance" experiment are shown in Figure B.2. The results on the graph on the left show that the difference between the NIST-SU error rate among the three classifiers for 1*k* of in-domain data is even bigger than in the case of BED, BMR and BRO (60.5% to 55.0% vs. 62.9% to 59.7%). Starting from 80*k* of in-domain data, the results of the three classifiers converge more consistently on Figure B.2 than on Figure B.1, and the difference is almost null between the performance of the three classifiers after 80*k*, whereas it was still around 1% on Figure B.1.

The comparison of BED experiment with the random experiment shows that the results on the BED experiment are not exactly the same as the ones on a random experiment. However, the differences between the performance of the classifiers on Figure B.1 are too small to allow us to affirm that the three meeting types BED, BMR and BRO require adaptation. The three meeting types are indeed not that different from a classification viewpoint, although they have different vocabularies and characteristics as shown in Tables 6.4 and 7.3.



(a) NIST-SU (b) F-Measure

Figure B.2: NIST-SU error rate (a) and F-Measure (b) for three randomly created training sets.



(a) NIST-SU (b) F-Measure

Figure B.3: NIST-SU error rate (a) and F-Measure (b) for the three meeting types BED, BMR and BRO evaluated on MRDA.

Bibliography

ALLEN, J. 1995. *Natural Language Understanding*. 2nd edn. Benjamin/Cummings Publ. 6

ANG, J., LIU, Y., & SHRIBERG, E. 2005. Automatic Dialog Act Segmentation and Classification in Multiparty Meetings. *In: Proceedings of ICASSP*. 41

BACCHIANI, M., & ROARK, B. 2003 (April). Unsupervised Language Model Adaptation. *In: Proceedings of the ICASSP*. 18

BACCHIANI, M., ROARK, B., & SARACLAR, M. 2004. Language Model Adaptation with MAP Estimation and the Perceptron Algorithm. *In: Proceedings of HLT-NAACL*. 18

BAKER, J. 1979. Trainable grammars for speech recognition. *Pages 547–550 of: Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*. 6

BLUM, A., & LANLGEY, P. 1997. Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*.

BOITE, R., BOURLARD, H., DUTOIT, T., HANCQ, J., & LEICH, H. 1999. *Traitement de la Parole (Speech Processing)*. Presses Polytechniques et Universitaires Romandes, Swiss Federal Institute of Technology Lausanne. 9

BRATT, H. 2006. Algemy, a tool for prosodic features analysis and extraction. *Personal Communication*. 37

BREIMAN, L. 1996. *Bias, variance and arcing classifiers*. Tech. rept. University of California Berkeley. 22

CETIN, O., & SHRIBERG, E. 2006. Overlap in Meetings: ASR Effects and Analysis by Dialog Factors, Speakers, and Collection Site. *In: roc. of MLMI*. 2, 6, 36

CHELBA, C., & ACERO, A. 2004. Adaptation of Maximum Entropy Capitalizer: Little Data Can Help a Lot. *In: Proceedings of EMNLP*. 19

DAUMÉ, H., & MARCU, D. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California, Los Angeles, CA. 15, 19

DHILLON, R., BHAGAT, S., CARVEY, H., & SHRIBERG, E. 2004. *Meeting Recorder Project: Dialog Act Labeling Guide*. Tech. rept. ICSI. 14, 32, 33

DIETTERICH, T. G. 2002. *Ensemble Learning*. Cambridge, MA: The MIT Press. Pages 405–408.

EISENSTEIN, J., & DAVIS, R. 2005. Gesture Features for Sentence Segmentation. *In: Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW'05)*. 12, 56

FLEURET, F. 2006. *Short Note: Boosting as a Gradient Descent*. Demonstration of Boosting being Gradient Descent algorithm. 24

FLORIAN, R., HASSAN, H., JING, H., KAMBHATLA, N., LUO, X., NICOLOV, N., & ROUKOS, S. 2004. A Statistical Model for Multilingual Entity Detection and Tracking. *In: Proceedings of the Human Language Technologies Conference 2004 (HLT-NAACL'04)*. 19

FREUND, Y., & SCHAPIRE, R. 1995. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*. 21

FRIEDMAN, J., HASTIE, T., & TIBSHIRANI, R. 1998. *Additive logistic regression: a statistical view of boosting*. 21, 24

GADDE, V., STOLCKE, A., VERGYRI, D., ZHENG, J., SONMEZ, K., & VENKATARAMAN, A. 2002. Building an ASR System for Noisy Environments: SRI's 2001 SPINE Evaluation System. *Pages 1577–1580 of: Proceedings of ICSLP*, vol. 3. 8

GODFREY, J. J., HOLLIMAN, E. C., & MCDANIEL, J. 1990. Switchboard: Telephone speech Corpus for Research and Development. *In: Proceedings of ICASSP*.

GORIN, A. L., RICCARDI, G., & WRIGHT, J. H. 1997. How may I help you? *Speech Communication*, **23**(1/2), 113–127. 1

GOTOH, Y., & RENALS, S. 2000. Sentence Boundary Detection in Broadcast Speech Transcripts. *In: Proceedings of ISCA ITRW Workshop*. 11

GUPTA, N., TUR, G., HAKKANI-TÜR, DILEK, BANGALORE, S., RICCARDI, G., & RAHIM, M. 2006. The AT&T Spoken Language Understanding System. *Pages 213–222 of: IEEE Transactions on Speech and Audio Processing*, vol. 14. 14

HILLARD, D., OSTENDORF, M., STOLCKE, A., LIU, Y., & SHRIBERG, E. 2004. Improving Automatic Sentence Boundary Detection with Confusion Networks. *In: Proceedings of HLT-NAACL*. 11

HUANG, X., ACERO, A., & HON, H. 2001. *Spoken Language Processing*. Prentice Hall. 18

JANIN, A., ANG, J., BHAGAT, S., DHILLON, R., EDWARDS, J., MACIAS-GUARASA, J., MORGAN, N., PESKIN, B., SHRIBERG, E., STOLCKE, A., WOOTERS, C., & WREDE, B. 2004. The ICSI Meeting Project: Resources and Research. *In: Proceedings of ICASSP*. 31

JURAFSKY, D. 1994. The Berkeley Restaurant Project. *In: Proceedings of ICSLP*. 1

JURAFSKY, D., & MARTIN, J. H. 2000. *Speech and Language Processing*. Prentice Hall. 7, 9

JURAFSKY, D., BATES, R., COCCARO, N., MARTIN, R., METEER, M., RIES, K., SHRIBERG, E., STOLCKE, A., TAYLOR, P., & ESS-DYKEMA, C. VAN. 1998. *Switchboard Discourse Language Modeling Project Report Research Note*. Tech. rept. Center for Speech and Language Processing, Johns Hopkins University. 34

KOLEN, M., & BRENNAN, R. 2004. *Test Equating, Scaling, and Linking: Methods and Practices*. Springer. Chap. 6, pages 177–180. 65

LIU, Y., SHRIBERG, E., STOLCKE, A., PESKIN, B., ANG, J., HILLARD, D., OSTENDORF, M., TOMALIN, M., WOODLAND, P., & HARPER, M. 2005. Structural Metadata Research in the EARS Program. *In: Proceedings of ICASSP*. 12

LIU, Y., SHRIBERG, E., STOLCKE, A., HILLARD, D., OSTENDORF, M., PESKIN, B., & HARPER, M. 2006. The ICSI-SRI Spring 2006 Meeting Recognition System. *In: Machine Learning for Multimodal Interaction: Third International Workshop*. 5, 12

MAKHOUL, J., BARON, A., BULYKO, I., NGUYEN, L., RAMSHAW, L., STALLARD, D., SCHWARTZ, R., & XIANG, B. 2005. The Effects of Speech Recognition and Punctuation on Information Extraction Performance. *In: Proceedings of Interspeech*. 5, 12

METEER, M., & IYER, R. 1996. Modeling conversational speech for speech recognition. *In: Proceedings of Conference on Empirical Methods in Natural Language Processing*. 11

MROZINSKI, J., WHITTAKER, E., CHATAIN, P., & FURUI, S. 2005. Automatic Sentence Segmentation of Speech for Automatic Summarization. *In: Proceedings of ICASSP*. 1, 5, 12

NETER, J., & WASSERMAN, W. 1974. *Applied Linear Statistical Models*. Richard D. Irwin, Inc. Chap. 6, 7, 8, 9. 65

REYZIN, L., & SCHAPIRE, R. 2006. How boosting the margin can also boost classifier complexity. *In: Proceedings of the 23rd Internation Conference on Machine Learning*. 22

ROARK, B., & BACCHIANI, M. 2003. Supervised and Unsupervised PCFG Adaptation to Novel Domains. *In: Proceedings of HLT-NAACL*. 18, 19

ROARK, B., LIU, Y., HARPER, M., STEWART, R., LEASE, M., SNOVER, M., SHAFRAN, I., DORR, B., HALE, J., KRASNYANSKAYA, A., & YUNG, L. 2006. Reranking for Sentence Boundary Detection in Conversational Speech. *In: Proceedings of ICASSP*. 11, 12

SCHAPIRE, R. 2001. The Boosting Approach to Machine Learning: An Overview. *In: Proceedings of MSRI Workshop on Nonlinear Estimation and Classification*. 22

SCHAPIRE, R., & SINGER, Y. 2000. Boostexter: A Boosting-based System for Text Categorization. *Machine Learning*, **39**(2/3). 16, 22

SCHAPIRE, R., FREUND, Y., BARTLETT, P., & LEE, W. 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*. 22

SCHAPIRE, R., ROCHERY, M., RAHIM, M., & GUPTA, N. 2005. Boosting with prior knowledge for call classification. *In: IEEE Transactions on Speech and Audio Processing*. 27

SHANNON, C. E. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal*, 379–423.

SHRIBERG, E. 2005. Spontaneous Speech: How People Really Talk, and Why Engineers Should Care. Proceedings of Eurospeech, Lisbon. *In: Proceedings of Eurospeech*. 2, 6

SHRIBERG, E., STOLCKE, A., HAKKANI-TÜR, D., & TUR, G. 2000. Prosody-Based Automatic Segmentation of Speech into Sentences and Topics. *Speech Communication*. 5, 11, 12

SHRIBERG, E., DHILLON, R., BHAGAT, S., ANG, J., & CARVEY, H. 2004. The ICSI Meeting Recorder Dialog Act (MRDA) Corpus. *In: Proceedings of SigDial Workshop*.

SOLTAU, H., KINGSBURY, B., MANGU, L., POVEY, D., SAON, G., & ZWEIG, G. 2005. The IBM 2004 Conversational Telephony System for Rich Transcription. *In: Proceedings of ICASSP*. 5

SOUVIGNIER, B., & KELLNER, A. 1998. Online Adaptation for Language Models in Spoken Dialogue Systems. *In: Proceedings of ICSLP*. 15

STOLCKE, A. 2002 (September). SRILM – An Extensible Language Modeling Toolkit. *In: Proceedings of the ICSLP*. 29

STOLCKE, A., & SHRIBERG, E. 1996. Automatic linguistic segmentation of conversational speech. *In: Proceedings of ICSLP*, vol. 2. 11

STOLCKE, A., ANGUERA, X., BOAKYE, K., CETIN, O., GRÉZL, F., JANIN, A., MANDAL, A., PESKIN, B., WOOTERS, C., & ZHENG, J. 2005. Further progress in meeting recognition: The ICSI-SRI Spring 2005 speech-to-text evaluation system. *In: Proceedings of MLMI*.

TUR, G. 2005. Model Adaptation for Spoken Language Understanding. *In: Proceedings of ICASSP*. 16, 19, 25, 27, 50, 53

TUR, G., & HAKKANI-TÜR, D. 2003. Exploiting Unlabeled Utterances for Spoken Language Understanding. *In: Proceedings of EUROSPEECH*.

TUR, G., GUZ, U., & HAKKANI-TÜR, D. 2006. Model Adaptation for Dialog Act Tagging. *In: Proceedings of SLT 2006*.

VON AHN, L. 2006. Games With A Purpose. *IEEE Computer Magazine*, 96–98. 2

X. FANG, J. GAO, J. LI, & SHENG, H. 2003. Training data optimization for language model adaptation. *In: Proceedings of Eurospeech*. 19

ZIMMERMANN, M., HAKKANI-TR̈, D., FUNG, J., MIRGHAFORI, N., E.SHRIBERG, & LIU, Y. 2006a. The ICSI+ Multi-Lingual Sentence Segmentation System. *In: Proceedings of ICSLP*. 11, 14, 16, 22

ZIMMERMANN, M., STOLCKE, A., & SHRIBERG, E. 2006b. Joint Segmentation and Classification of Dialog Acts in Multiparty Meetings. *In: Proceedings of ICASSP*. 7, 41