Karlsruhe Institute of Technology

Speech Recognition on English-Mandarin Code-Switching Data using Factored Language Models

- with Part-of-Speech Tags, Language ID and Code-Switch Point Probability as Factors

Diplomarbeit at the Cognitive Systems Lab
Prof. Dr.-Ing. Tanja Schultz
Department of Informatics
Karlsruhe Institute of Technology

and at the Human Language Technology Center
Dr. Pascale Fung, Associate Professor
Department of Electronics and Computer Engineering
Hong Kong University of Science and Technology

by

cand. inform.
Jan Gebhardt

Supervisors:

Prof. Dr.-Ing. Tanja Schultz
Dr. Pascale Fung, Associate Professor
Dipl. Inform. Tim Schlippe

Tag der Anmeldung: 1. Oktober 2010
Tag der Abgabe: 31. März 2011

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

21. März 2011

Abstract

Code-switching is defined as "the alternate use of two or more languages in the same utterance or conversation" [1]. CS is a wide-spread phenomenon in multilingual communities, where multiple languages are concurrently used in a conversation.

For automatic speech recognition (ASR), particularly intra-sentential code-switching poses an interesting challenge due to the multilingual context for language modeling and the acoustic model. The statistical estimation of n-grams at code-switch points (CSP) is poor for common, word-based n-gram language models.

Our work investigates the application of additional features for ASR on Mandarin-English speech data containing intra-sentential CS.

First we explore the use of various features for multilingual language modeling by predicting language and CSPs. We investigate the text based prediction of the language of a word, utilizing language, part-of-speech (POS) tags and occurrence features of previous words in the current utterance. The results of our language prediction experiments are used to predict CSPs. We can reach an improvement compared to random predictions of CSPs. The results of our experiments to predict language and CSPs indicate, that it is useful to add language and POS tag features in the multilingual language model applied on CS data.

Consequently, we include language identification (LID), POS tags, and a CSP probability class into the language model. We utilize factored language models to incorporate these features. N-best rescoring is applied to use factored language models for speech recognition. We obtain the highest performance utilizing factored language models with LID, POS tags and words as features. We evaluate our results utilizing a mixed error rate (MER). The MER is defined as the character error rate (CER) for Mandarin and the word error rate (WER) for English. Using our best performing factored language model, we improve the MER performance from 59.1% to 58.4% on the development set and from 60.8% to 59.8% on the evaluation set.

Acknowledgements

First, I would like to thank Prof. Tanja Schultz for providing me the opportunity to conduct research abroad at the Hong Kong University of Science and Technology and supervising my work at the Cognitive Systems Lab.

Furthermore I would like to thank Prof. Pascale Fung and Dipl. Inform. Tim Schlippe for supervising my work and supporting me with their advice, ideas and helping me with problems.

Also I would like to thank Ying Li, Ho Yin Chan, Jian Zhang, Ying Dou, Thang Vu, Fabian Blaicher, Dominic Telaar, Franziska Kraus, Edy Guevara Komgang and Sebastian Ochs for their support.

My research at the Hong Kong University of Science and Technology was supported by the Baden-Württemberg-Stipendium (interACT) granted by the Landesstiftung Baden-Württemberg.

Zusammenfassung

Code-switching (CS) ist definiert als die Verwendung von mehr als einer Sprache innerhalb eines Satzes oder einer Konversation. CS ist ein weit verbreitetes Phänomen in mehrsprachigen Gemeinschaften, in denen mehrere Sprachen gleichzeitig in Konversationen verwendet werden.

Für die automatische Spracherkennung (ASR) stellt insbesondere intraphrasales Code-Switching aufgrund des Sprachwechsels innerhalb von Sätzen eine interessante Herausforderung dar. Die statistische Schätzung von N-Grammen mit Sprachwechsel in normalen N-Gram Sprachmodellen ist schwierig.

In unserer Arbeit wird der Nutzen von zusätzlichen Merkmalen für ASR auf chinesisch-englischen Sprachdaten mit intraphrasalem CS untersucht.

Um den Nutzen von verschiedenen Merkmalen für multilinguale Sprachmodellierung zu untersuchen, wurde die Sprache von Wörtern und die Punkte für Sprachwechsel (CSP) prädiziert. Wir bewerten unsere Ergebnisse auf dem Development Set und dem Evaluation Set. Das Development Set wird verwendet um geeignete Klassifikatoren für unsere Prädiktionsexperimente auszuwählen, Designs für faktorierte Sprachmodelle (FLM) zu evaluieren und ASR Parameter zu bewerten. Die textbasierte Vorhersage der Sprache eines Wortes wird basierend auf der Sprache, Part-of-speech (POS) Tag und der Anzahl vorheriger Worte in dem aktuellen Satz durchgeführt. Dabei erreichen wir ein F-Measure von 0,926 auf unserem Development Set und 0,901 auf unserem Evaluation Set. Basierend auf unseren Experimenten zur Vorhersage der LID prädizieren wir CSPs. Wir erreichen ein F-Measure von 0,205 auf unserem Development Set und 0,300 auf unserem Evaluation Set. Wir konnten uns gegenüber einer zufälligen Prädiktion von CSPs verbessern, da bei der zufälligen Prädiktion nur ein F-Measure von 0,116 auf dem Development Set und 0,142 auf dem Evaluation Set erreicht wurde. Wir stellen also fest, dass LID und POS Tag nützliche Informationen für das Sprachmodell bei CS enthalten.

Der Fokus unserer Arbeit liegt in der Integration und Bewertung von POS Tags, LID und einer klassifizierten CSP Wahrscheinlichkeit in das Sprachmodell. Wir bestimmen die POS Tags aus unseren Textdaten unter Verwendung von POS Taggern. Die LID wird regelbasiert aus den Texten abgeleitet. Aufgrund der unterschiedlichen Schriften von Chinesisch und Englisch, kann die LID fehlerfrei bestimmt werden. Die CSP Klassen werden basierend darauf, ob nach dem aktuellem Wort ein CSP folgt, bestimmt. Die Einschätzung der CSP Wahrscheinlichkeiten erfolgt auf unseren Trainingsdaten. FLMs werden verwendet um LID, POS Tags und CSP Wahrscheinlichkeiten in unseren Sprachmodell zu integrieren und ein N-best Rescoring durchzuführen. Die höchste Verbesserung erreichen wir mit FLMs die neben Wörtern, LID und POS Tags beinhalten. Zur Bewertung unserer Ergebnisse verwenden wir eine Mischfehlerrate (MER), die sich aus der Wortfehlerrate für Englisch und aus der Zeichenfehlerrate für Chinesisch zusammensetzt. Mit unserem FLM mit der besten Performance, können wir die MER auf dem Development Set von 59,1% auf 58,4% und auf dem Evaluation Set von 60,8% auf 59,8% reduzieren.

Contents

1. Introduction

With the ongoing globalization, the contact between people with different origins and mother tongues increases. Particular in communities, where a significant share of people from foreign origin work, multiple languages are present. Changing the language in conversations is very common in informal settings for multilingual speakers [2]. For example in Hong Kong where Chinese (Cantonese and Mandarin) and English are spoken languages, code-switching is a wide-spread phenomenon. Code-switching is defined as "the alternate use of two or more languages in the same utterance or conversation" [1]. It can happen inside utterances or at their boundaries. Code-switching inside utterances is referred to as intra-sentential code-switching while code-switching at utterance boundaries is called inter-sentential code-switching.

In the context of automatic speech recognition, code-switching poses an interesting challenge. Code-switching leads to increased complexity considering resources like the vocabulary and dictionary. In particular the statistical estimation of n-grams is poor for common, word-based n-gram language models.

1.1 Goal

The goal of our work is to investigate means to add information for speech recognition, which may be useful for recognizing speech recognition with code-switching inside utterance.

We perform experiments with text-based prediction of language and code-switching points based upon language identification, part of speech tag and occurrence features of preceding words in the current utterance. With these prediction experiments, we investigate if language identification and part-of-speech tag features may contain useful features for recognizing utterances with intra-sentential code-switching. Our main focus is to integrate language identification, part of speech tag, and code-switch point probabilities in the multilingual language model for speech recognition. We utilize factored language models [3] for this task.

1.2 Structure of our Work

Chapter 2 describes the fundamentals of code-switching including attitudes and reasons for code-switching as well as the mechanics for code-switching. The basics of speech recognition are also described in this chapter. Additionally, we provide information about code-switching and speech recognition, especially considering related work. Afterwards we describe evaluation methods for classification which we need for further experiments.

Chapter 3 contains the description of our Chinese-English database and tools we applied for this work. We describe the properties of the database and our split in training, development and evaluation set.

Chapter 4 is the main part of this work. This chapter consists of

- Part-of-speech tagging for English-Mandarin data containing intra-sentential code-switching.
- Text based prediction of language identification and code-switching points.
- Our baseline speech recognition system.
- Fundamentals about factored language models.
- Our experiments to utilize factored language models for recognizing utterances with intra-sentential code-switching.

Chapter 5 is a summary of our work.

2. Fundamentals

2.1 Code-Switching

Changing the language in conversations is very common in informal settings for multilingual speakers [2]. Code-switching is defined as

- "the alternate use of two or more languages in the same utterance or conversation" [1].

Another definition for code-switching by [4] defines it as:

- "the juxtaposition within the same speech exchange of passages of speech belonging to two different grammatical systems or subsystems."

An example of French-English code-switching [1]:

- "Va chercher Marc (go fetch Marc) *and bribe him* avec un chocolat chaud (with a hot chocolate) *with cream on top.*"

Another example for code-switching in Spanish-English [1]:

- "No me fije hasta que ya no me dijo (I didn't notice he told me): *Oh I didn't think he'd be there.*

The difference in these two examples is that in the first example code-switching happens inside the sentence, while in the second example code-switching happens at sentence boundaries. Therefore the majority of the authors divide code-switching into two types:

- Inter-sentential code-switching: code-switching at utterance or sentence boundaries.

- Intra-sentential code-switching: code-switching inside an utterance or a sentence.

Inter-sentential code-switching is a special case of intra-sentential code-switching.
For code-switching, the matrix language and the embedded language are important terms [5]:
The matrix language for an utterance is the language, where the majority of words in the utterance belongs to. For code-switching involving two languages, like in our case, all words in an utterance, which do not belong to the matrix language, belong to the embedded language.
Code-switching should not be confused with borrowing, where foreign words are included and integrated morphologically and phonologically into the local language.

2.1.1 Attitudes toward Code-Switching

An interesting field is the perception of code-switching by the people, who use code-switching. Grosjean [1] describes that monolinguals had a negative attitude towards code-switching for a long time, since it was seen as a grammar-lacking mixture of two languages. People who used code-switching extensively were often said to know none of the languages they speak well enough to converse in one language alone. Monolinguals seem to have a negative attitude towards code-switching. Consequently, some multilinguals restrict code-switching to situations where they will not be stigmatized for switching languages. However, code-switching is a very useful communication phenomenon. It often happens unconsciously and speakers may be unaware that they code-switch. The main objective for a (multilingual) speaker is that the recipients get the message and intent of the speaker.

2.1.2 Reasons for Code-Switching

Grojean [1] wrote that bilinguals say that the reason for code-switch is the lack of facility in one language for a particular topic. Frequently, bilinguals know the words in both languages, but a word may be more available in the language they currently do not speak. He provides a list of reasons for code-switching:

- Fill a linguistic need for a lexical item, set phrase, discourse marker or sentence filler.
- Continue the last language used (triggering).
- Quote someone.
- Qualify message: amplify or emphasize (*topper* in argument).
- Specify speaker involvement (personalize message).
- Mark and emphasize group identity (solidarity).
- Convey confidentiality, anger and annoyance.
- Exclude someone from conversation.
- Change role of speaker, raise status, add authority, show expertise.

2.1.3 Mechanics of Code-Switching

This subsection deals with the position of intra-sentential code-switch points. The research for intra-sentential code-switching has focused on syntax. Linguists proposed several constraints for code-switching. The most common ones are described below [6]:

- Free-morpheme constraint: "A switch may not occur between a bound morpheme and a lexical form unless the latter has been phonetically integrated into the language of the bound phoneme". A bound morpheme is a morpheme, which exclusively appears as a part of a longer word and cannot stand alone [7].
- Equivalence constraint: "the order of sentence constituents immediately adjacent to and on both sides of the switch point must be grammatical with respect to both languages involved simultaneously".

All of the proposed constraints are controversial as linguists have offered apparent counter examples [8] [9]. Therefore we did not consider any of these constraints in our work. We investigate the use of different features for prediction and language modeling using data containing code-switching.

2.2 Automatic Speech Recognition Basics

Automatic speech recognition (ASR) aims to generate the accurate written transcription from the spoken utterance [10]. According to [11], we apply digital signal processing and feature extraction to the input speech which will provide a sequence of acoustic vectors $Y = y_1, y_2, ...$ An utterance consists of a word sequence $W = w_1, w_2, ..., w_m$. Our goal is to obtain the most likely word sequence $\widehat{W}$ corresponding to the spoken utterance which is calculated using the fundamental equation of speech recognition:

$$\widehat{W} = argmax_W P(W|Y) = argmax_W \frac{P(W) \cdot P(Y|W)}{P(Y)} \tag{2.1}$$

$P(Y)$ does not need to be calculated since $argmax_W$ is the maximum of the equation for all possible word sequences W and $P(Y)$ does not depend on W. $P(W)$ is calculated in the language model. $P(Y|W)$ is provided in the acoustic model and the pronunciation dictionary, and $argmax_W$ is represented during the search (decoding).

Our work focuses on the language model. We investigate the benefit from additional information in the language model for recognizing utterances containing code-switching.

2.2.1 Language Model

There are two common types of language models: Statistical language models and grammars. Grammars are useful for very limited domains. They consists of a set of rules and a vocabulary. More about grammars can be found in [12]. Since we are

working with utterances from an unrestricted domain, we are working with statistical language models.
The statistical language model assigns the probability $P(W)$ to a word sequence. For example, in English the word sequence: "Dear ladies and gentlemen" is much more likely than "Deer lay dee's sand dental man" but the sound of these two word sequences is very similar. When calculating

$$P(W) = \prod_{i=1}^{k} P(w_i|\ w_{i-1}...w_2\ w_1) \tag{2.2}$$

for a large vocabulary v, there is a huge number of possible $|v|^k$ word sequences. Therefore, the n-gram technique is used:

$$P(W)_{n-gram} = \prod_{i=1}^{k} P(w_i|\ w_{i-1}...\ w_{i-n+1}) \tag{2.3}$$

where often $n = 3$ [12].

The frequency of the occurrence of a word $P(w_i|history)$ can be estimated by the count $C(w_i|history)$ divided by the count of the history. For a 4-gram language model the probability of a word depends on its three preceding words and can be estimated as follows:

$$P(w_i|w_{i-3}w_{i-2}w_{i-1}) = \frac{C(w_{i-3}w_{i-2}w_{i-1}w_i)}{C(w_{i-3}w_{i-2}w_{i-1})} \tag{2.4}$$

For building an n-gram language model, it is best to have a huge training corpus with millions of words available. However, for code-switching data we only have a very small amount of data available because code-switching is frequently used in (spontaneous) speech but not in (formal) writing.

A key problem in n-gram modeling is data sparseness. Many possible word combinations are not observed in the training. Given we have an unobserved word combination in the test data, the probability of a string W containing an unobserved word combination is 0. This is obviously an underestimation. If $P(W) = 0$, the string is not considered as a possible transcription, although it may be the accurate transcription. The goal of n-gram smoothing is to assign non-zero probabilities to all word n-grams that did not appear in the training data. Smoothing adjusts probabilities of word sequences to get a more robust estimation for unseen data. It tends to improve the accuracy of the model as a whole. There are various smoothing techniques described in [12]. Backoff smoothing is a technique, where if a higher order n-gram $w|w_{i-n+1}\cdots w_{i-2}w_{i-1}$ has a non-zero probability, this distribution is used. If the count of the higher order n-gram is zero, we backoff to the lower order n-gram $w|w_{i-n+2}\cdots w_{i-2}w_{i-1}$. Kneser-Ney Smoothing is a popular smoothing method which we use in our experiments.

Important language model measures are the perplexity and the out-of-vocabulary (OOV) rate. The perplexity is the average branching factor of an n-gram language model. On a given domain and vocabulary a language model with a low perplexity should be more suitable than a language model with a higher perplexity [13]. The

OOV is the frequency of (word) tokens inside the test corpus, which are not in the vocabulary of the language model. These words are mapped to an unknown word and get an equal probability. A high OOV rate may lead to poor speech recognition results as OOV words can at best be estimated as unknown words by the language. This will definitely lead to a recognition error for the current word and might lead to consecutive errors due to a wrong context.

It is possible to combine different language models or text corpora. If more than one corpus is available, these corpora normally do not have an equal importance for the speech data to recognize. In such cases language model interpolation is a common approach [14]. The simplest way of language model combination is the linear interpolation. Provided we have the language model or text corpus a and the language model or text corpus b, then the probability $Pr(.,.)$ of a word w_q given a history h_q for the interpolated language model can be calculated in the following way:

$$Pr(w_q|h_q) = (1-\lambda)Pr_a(w_q|h_q) + \lambda Pr_b(w_q|h_q),\ 0 \leq \lambda \leq 1 \tag{2.5}$$

where λ is the interpolation coefficient. This coefficient can be calculated automatically by minimizing the perplexity on a development set as for example implemented in the SRI Language Model Toolkit [15].

2.2.2 Mixed Error Rate

The performance of a speech recognition system is measured by the word error rate (WER). The WER is calculated in the following way:

$$WER = \frac{INS + DEL + SUB}{N} \cdot 100\% \tag{2.6}$$

where INS is the number of word insertions, DEL the number of word deletions, SUB the number of word substitutions and N the number of words in the reference. Note that for the WER the best alignment should always be used. For example, if we have the reference: "Multilingual speech recognition is very interesting". Our speech recognizer gives the transcription "'Multi label beach recognition is very interesting"' the WER ist $\frac{1+0+2}{6} \cdot 100\% = 50\%$. The WER is an important indicator how good a speech recognizer is.

For English data normally the WER is estimated, while for Chinese data the character error rate (CER) is standard, since Chinese is lacking an explicit word segmentation. Because we have mixed Chinese-English data, we evaluate the WER on the English part and the CER on the Chinese part. For this work, we refer to this error rate on Chinese characters and English words as mixed error rate (MER). The MER is calculated in the same way as the WER, but for the Chinese part characters are considered instead of words. We calculate the mixed error rate for our Chinese-English data in the following way:

$$MER = \frac{INS_m + DEL_m + SUB_m}{N_m} \cdot 100\% \tag{2.7}$$

where

- INS_m is the number of English word and Chinese character insertions,
- DEL_m is the number of English word and Chinese character deletions,
- SUB_m is the number of English word and Chinese character substitutions,
- N_m is the number of English words and Chinese characters in the reference.

2.3 Related Work

Multilingual speech recognizers are able to process multiple languages and can be created by merging components of monolingual speech recognizers. In this work, we define a speech recognizer as multilingual if at least the acoustic model, the language model or pronunciation dictionary is used and trained from more than one language [16].

For recognizing code-switched utterances, we need to be able to handle speech in all languages involved in code-switching.

A common approach to handle inter-sentential code-switching is described in [2]. It uses finite state grammars and works in the following way:

1. Define a distinct vocabulary Σ_i for every language i we consider.
2. Train a finite state acceptor G_i which accepts Σ_i^*.
3. The multilingual language model is the finite state union: $\bigcup_{i=1}^{M} G_i$.

A regular language is a formal languages, which can be accepted by a finite state machine. Regular languages are closed on union operations. Therefore, the result of $\bigcup_{i=1}^{M} G_i$ is an n-gram language model. This approach has the advantage that powerful algorithms for finite state machines can be applied to handle the n-gram models.

Intra-sentential code-switching is very common in informal settings for multilingual speakers. Two techniques have been investigated for modeling such multilingual speech:

- Merge the language model training texts with possible tagging of words, which have the same spelling in more than one of the considered languages, to be able to distinguish these words and build a language model of the merged training texts. For example in German the word “beamer” stands for a data projector, while in English “beamer” is slang for a car of the company BWM. Therefore the word “beamer” needs to be tagged, if we would be interested in English and German.
- Merge the vocabulary of multiple languages but estimate an n-gram language models for each individual language. These language models are interpolated afterwards. Word entries of one language are included in the language models of other languages with zero counts and received a small backoff probability. In this approach, all words have non-zero probabilities, which enables the approach to handle intra-sentential code-switching.

The performance on intra-sentential code-switching tends to be worse than on inter-sentential code-switching. The word data merging approach tends to be significantly worse than model interpolation [2].

Lyu and Lyu [17] proposed a two stage framework for speech recognition on Mandarin-Taiwanese code-switching speech data:

1. LVCSR-based language identification.
2. Language dependent speech recognition.

They use cues to distinguish Mandarin and Taiwanese in the language identification component. Afterwards they apply different syllables as phonetic cues, tones as prosodic cues, and rhythm cues as well as the variation of the duration of a series of sound. To connect acoustic model, duration model and language model, a maximum a posteriori decision rule is used. The authors are able to reduce the error rate by 34.5% for language identification and the syllable error rate by 17.7% (from 44.7% to 36.8%) for automatic speech recognition compared to their baseline.

The following papers are related to our work, but were not published at the beginning of our work. Therefore experiments described in these papers could not be considered for our experimental setup.

Yeh et al. proposed a paper about an integrated framework for transcribing Mandarin-English code-mixed lectures with improved acoustic and language modeling [18]. Their speech data is only from a single speaker. Yeh et al. use class based n-grams for statistical language modeling. They cluster words into classes based upon part-of-speech features and a perplexity criterion. Additionally, the authors use random forest language models and language model adaptation. Yeh et al. achieve the highest improvement of 1.46% relative using class-based language models and part of speech tags.

Cao et al. proposed a paper about semantic-based language modeling for Cantonese-English code-mixing speech recognition [19]. To deal with the lack of training data for code-mixed language modeling, the authors use a translation based mapping scheme. Embedded English word are clustered into about 200 semantics classes differentiated with the word meaning, part-of-speech and syntactic function. The performance is evaluated on a Cantonese-English read speech corpus. Similar to our MER calculation the error rate is in terms of CER for Cantonese and in terms of WER for English. Cao et al. obtain an overall error rate of 26.3%, while their baseline 3-gram language model achieved an error rate of 29.5%. In particular, they achieve an improvement for English words where the error rate is reduced from 51.4% to 36.1%.

Tsai et al. studied on Hakka and mixed Hakka-Mandarin speech recognition [20]. They apply different methods to incorporate part-of-speech and word translation information into the language model. The authors use a class-based language model to add part-of-speech information into the language model.

Bhuvanagiri et al. proposed an approach to mixed language automatic speech recognition [21]. The authors claim to achieved a good recognition performance with applying a language model, which was built on a small mixed speech corpus, and constructing a pronunciation dictionary for mixed language words.

2.4 Evaluation Measures for Classification

In our experiments to predict language and code-switch points, we utilize classification. We describe general classification measure in this section.

We define the (desired) class, we are interested in, as the positive class. Assuming we have a two-class problem, there are two possible predictions of the classifier and two possible results [22]. Given we have the classes *1* and *0*, where class *1* is the positive class, there are four possible combinations [22]:

- **True positive (TP):** The predicted class and the actual class are the positive class *1*.
- **True negative (TN):** The predicted class and the actual class are the negative class *0*.
- **False positive (FP):** The predicted class is the positive class *1*, while the actual class is *0*.
- **False negative (FN):** The predicted class is the negative class *0*, while the actual class is *1*.

An example for classification may be the evaluation of the mood of a person which we can classify into positive *happy* and negative *unhappy*. If we predict the person to be *happy* and the person actually is *happy* we have a TP. Given we correctly predict a person to be *unhappy* this is a TN. If the prediction is that the person is *happy* while the person is actually *unhappy* we create a FP. Likewise a prediction for a person to be *unhappy* while the person actually is *happy* is a FN.

From true positives, true negatives, false positives and false negatives we can derive the measures recall, precision, F-measure, and success rate for evaluation [22]:

- **Recall:** $Recall = \frac{TP}{TP+FN}$
- **Precision:** $Precision = \frac{TP}{TP+FP}$
- **F-measure:** $F - measure = \frac{2 \cdot recall \cdot precision}{recall+precision}$
- **Success rate:** $Success\ rate = \frac{TP+TN}{TP+TN+FP+FN}$

As in WEKA [23] we also refer to the success rate as the correct instances.

3. Database and Tools

3.1 Mandarin-English Code-Switching Database

Speech data which contains code-switching is hard to find. We applied English-Mandarin data, which was recorded at the Human Language Technology Center at the Hong Kong University of Technology. The data is described in detail in [24]. The speech data is conversational meeting data and contains a large variety of topics including university administration, history and politics. The total length of the audio data is 163 minutes. While the speech data consists of different speakers, there is one dominant (female) speaker making up for 159 minutes of data. Table 3.1 gives a break down.

3.1.1 Database Split

For training, tuning and evaluating our ASR systems, we split the database into the following sets:

- **Training set:** The training set is used to train our models.
- **Development set:** The development set is applied to set suitable ASR parameters, N-best rescoring parameters, and to select classifiers for our prediction experiments.
- **Evaluation (test) set:** The evaluation (test) set is used for the final evaluation.

The original distribution of the complete data into training set, development set, and evaluation set by Burgmer [24] and others was processed session-based, where a whole session or part of a session founds the development and evaluation set. We evaluated this data division. Table 3.1 displays utterances (Utts) and their distribution into pure Chinese (CH) utterances, pure English (EN) utterances and CS utterances in the training set, development (dev.) set and evaluation (eval.) set. Table 3.2 contains CS points, perplexity and OOV-rate on a 4-gram language model

Data Set	Total Utts	CH Utts	EN Utts	CS Utts	% of CS Utts
Training	1017	428	124	465	46%
Dev.	267	146	28	93	35%
Eval.	103	43	27	33	32%

Table 3.1: Utterances(Utts) and their proportion of pure CH utterances, pure EN utterances and CS utterances by Burgmer [24].

Data Set	CS Points	CS Points per CS Utt	Perplexity	OOV
Training	1370	2.95	27.6	0.00%
Development	192	2.06	436.4	16.17%
Evaluation	98	2.97	519.3	16.76%

Table 3.2: CS points, perplexity and OOV-rate on a 4-gram language model built exclusively on the training set in the original distribution by Burgmer [24].

built exclusively on the training set in the different sets in the original distribution by Burgmer and others. While it is uncommon to use 4-gram language models for small data sets, experiments described in section 4.3 indicate that we can achieve a better performance applying 4-gram language models instead of 3-grams. It can be observed that the OOV rate is very high and that the development set has more than twice as many utterances than the evaluation set. The high OOV can be explained by the distribution of the utterances in different sets. The original distribution was split session based. The topics in the different sessions are to a certain extend different. Also the variation of the size of development and evaluation set can be explained due to the session-based split. Due to these problems, we split the data differently:
To reduce the OOV rate, we applied a different approach for splitting the whole speech database into the training, development and evaluation set. We consider all utterances in chronological order and for every group of ten utterances. We assign eight utterances to the training set, one to the development set and one to the evaluation set. Consequently, each set gets a part of each session and we reach the standard 80% , 10% , 10% distribution for training, development and evaluation set. To compare with the original statistics, we obtain the statistics shown in Table 3.3 and Table 3.4 for our new data divisions.

Data Set	Total Utts	CH Utts	EN Utts	CS Utts	% of CS Utts
Training	1111	492	141	478	43%
Dev.	138	62	20	56	41%
Eval.	138	63	18	57	41%

Table 3.3: Utterances (Utts) and their distribution into pure CH utterances, pure EN utterances and CS utterances set in our distribution.

Data Set	CS Points	CS Points per CS Utt	Perplexity	OOV
Training	1388	2.90	28.7	0.00%
Development	128	2.29	384.2	7.94%
Evaluation	144	2.53	405.7	8.20%

Table 3.4: CS points, perplexity and OOV-rate on a 4-gram language model built exclusively on the training set in our distribution.

Data Set	Total Utts	CH Utts	EN Utts	CS Utts	% of CS Utts
Training	1111	492	141	478	43%
Dev. (NF)	114	54	17	43	38%
Eval. (NF)	116	58	16	42	36%

Table 3.5: Utterances (Utts) and their distribution into pure CH utterances, pure EN utterances and CS utterances in our distribution without fillers (NF).

Table 3.3 and 3.4 show that our new distributions reduce the OOV rate by more than 50% relative, while reducing the perplexity of the development and evaluation set on the language model built on the training set. Also, the proportions of utterances and CS are more balanced. This indicates that development set and evaluation set represent the training data better. Therefore it is more suitable for our speech recognition experiments.
Another issue with the data is that our utterances contain fillers. Fillers can be described as (non-)words and are difficult to attribute to any language [24]. The utterances with fillers may be of poor quality. We generated a new database to investigate the effect of fillers by comparing this database with our previous one. The focus of our work was to investigate CS in clear speech and the filler problem is a different topic. Therefore we removed the utterances containing fillers from our development and evaluation set. The utterances with fillers may still be useful for speaker adaptation, which we use our training set for in speech recognition experiments, while the utterances with fillers tend to be difficult to recognize. The statistics of the new database are shown in Table 3.5 and 3.6. Table 3.6 shows the code-switch points, the average amount of code-switches per utterance containing code-switches, and the perplexity and OOV on a 4-gram language model exclusively built on the training data. Please note that the average amount of words per utterance is 13.4 on the training set, 13.6 on the development set and 11.8 on the evaluation set respectively. The training set contains 14983 word tokens, the development set 1546 word tokens and the evaluation set 1364 word tokens respectively. Removing utterances with fillers leads to a small reduction in OOV rate and to a 13.5% and 16.7% reduction in perplexity on the development and evaluation set. Due to these improvements, we use our data distribution without fillers in development and evaluation set for our prediction and speech recognition experiments, described in sections 4.2 and 4.5.

Data Set	CS Points	CS Points per CS Utt	Perplexity	OOV
Training	1388	2.90	28.7	0.00%
Dev. (no filler)	94	2.18	332.0	7.76%
Eval. (no filler)	103	2.45	337.9	8.14%

Table 3.6: Code-switch (CS) points, perplexity and OOV-rate on a 4-gram language model built exclusively on the training set in our distribution without fillers.

3.2 Waikato Environment for Knowledge Analysis

The Waikato Environment for Knowledge Analysis (WEKA) [23] was designed as a toolbox for machine learning and data mining. WEKA provides several graphical user interfaces (GUI), of which we only used the WEKA Explorer GUI. The WEKA Explorer GUI provides the following functionalities:

- Data can be loaded and converted with preprocessing tools.
- Classification and regression can be done with a wide variety of implemented algorithms.
- Clustering methods and algorithms are provided.
- Methods for associated rule mining are provided.
- WEKA provides attribute selection tools which are utilized to figure out which attributes are most useful.
- Data visualization is implemented.

WEKA is available for non-commercial use for free under the Gnu general public license (GPL).

3.3 SRI Language Modeling Toolkit

The SRI Language Modeling Toolkit (SRILM) [15] is a popular tool for statistical language modeling. It consists of a collection of C++ libaries, programs and helper scripts to support the creation and evaluation of language models.
SRILM was designed with the following goals:

- Reliable and efficient implementation of recent LM algorithms to support system development, in particular for speech recognition.
- Flexibility and the possibly to extend SRILM was a key goal to support research for new language model types.
- A rational and clean design was intended with providing an application programming interface and a toolbox for commands.

SRILM provides common language model operations for building and evaluating language models. Various types of language model types are supported. Recent versions support factored language models [3], which we apply for our experiments. Also related tasks to language modeling are supported, like N-best lists and word lattices.
SRILM is available for free for noncommercial use under an Open Source Community License.

3.4 Janus Recognition Toolkit

The Janus Recognition Toolkit (JRTK) [25] is a toolkit for automatic speech recognition. JRTK was developed by the Interactive Systems Laboratories at Carnegie Mellon University, USA and University of Karlsruhe, Germany. The toolkit implements an object-oriented approach where tcl scripts based environments allow the construction of different recognizers. JRTK is a programmable shell with transparent and efficient objects. JRTK provides objects for a wide variety of recognition approaches.

3.5 Part-of-Speech Tagger

Part-of-speech (POS) tagging is the process of aligning words to word types. The word itself and the context is considered. POS taggers perform POS tagging. For our work we apply Chinese and English Stanford Loglinear POS taggers [26] [27]. The utilized taggers are maximum-entropy based. The used English POS tagger has an accuracy of 96.97% on English Wall Street Journal 19-21 data, where 89.03% of unknown words are tagged correctly. The utilized Chinese tagger achieves an accuracy of 93.60% on a combination of Chinese and Hong Kong texts, where 81.61% of unknown words are tagged correctly [28].

4. Experiments

4.1 Part-of-Speech Tagging

We have Chinese-English text data containing intra-sentential code-switching (CS). There are monolingual POS taggers available for different languages, also for Mandarin and English. A bilingual Mandarin-English POS tagger would be most suitable for our case, since it can internally consider language switches and provide accurate tags for Mandarin and English. However, to our best knowledge only monolingual taggers exists so far. Therefore, we applied a language islands based approach by Burgmer [24] which works in the following way:

- Determine the matrix language for each utterance.
- Search the utterance for language islands, consisting of at least three consecutive words of the embedded language in the current utterance.
- Apply the POS tagger in the matrix language for all words, which are not language islands as defined above.
- Utilize the POS tagger in the embedded language for language islands.
- Less than three consecutive words of the embedded language are processed by the POS tagger in the matrix language of the current utterance.

This language island approach limits the amount of consecutive foreign words, a tagger tags, while maintaining context information as long as the number of consecutive words in the embedded language are less than three. It is possible to change the count of consecutive words to be considered a language island. For our experiments, a change of this parameter did not provide improvements with regards to MER applying factored language models described in section 4.5.

We apply the Stanford Log-linear Part-Of-Speech Taggers [26][27] for Mandarin and English. The Mandarin POS tagger uses the Penn Chinese Treebank tag set [30], while the English POS tagger uses the English Penn Treebank tag set [29]. The

1. CC	Coordinating conjunction	25. TO	*to*
2. CD	Cardinal number	26. UH	Interjection
3. DT	Determiner	27. VB	Verb, base form
4. EX	Existential *there*	28. VBD	Verb, past tense
5. FW	Foreign word	29. VBG	Verb, gerund/present participle
6. IN	Preposition/subordinating conjunction	30. VBN	Verb, past participle
7. JJ	Adjective	31. VBP	Verb, non-3rd ps. sing. present
8. JJR	Adjective, comparative	32. VBZ	Verb, 3rd ps. sing. present
9. JJS	Adjective, superlative	33. WDT	*wh*-determiner
10. LS	List item marker	34. WP	*wh*-pronoun
11. MD	Modal	35. WP$	Possessive *wh*-pronoun
12. NN	Noun, singular or mass	36. WRB	*wh*-adverb
13. NNS	Noun, plural	37. #	Pound sign
14. NNP	Proper noun, singular	38. $	Dollar sign
15. NNPS	Proper noun, plural	39. .	Sentence-final punctuation
16. PDT	Predeterminer	40. ,	Comma
17. POS	Possessive ending	41. :	Colon, semi-colon
18. PRP	Personal pronoun	42. (	Left bracket character
19. PP$	Possessive pronoun	43.)	Right bracket character
20. RB	Adverb	44. "	Straight double quote
21. RBR	Adverb, comparative	45. ‘	Left open single quote
22. RBS	Adverb, superlative	46. “	Left open double quote
23. RP	Particle	47. ’	Right close single quote
24. SYM	Symbol (mathematical or scientific)	48. ”	Right close double quote

Figure 4.1: The English PENN Treebank tag set [29]

AD	adverb	还
AS	aspect marker	着
BA	把 in ba-construction	把, 将
CC	coordinating conjunction	和
CD	cardinal number	一百
CS	subordinating conjunction	虽然
DEC	的 in a relative-clause	的
DEG	associative 的	的
DER	得 in V-de const. and V-de-R	得
DEV	地 before VP	地
DT	determiner	这
ETC	for words 等, 等等	等, 等等
FW	foreign words	I S O
IJ	interjection	啊
JJ	other noun-modifier	男, 共同
LB	被 in long bei-const	被, 给
LC	localizer	里
M	measure word	个
MSP	other particle	所
NN	common noun	书
NR	proper noun	美国
NT	temporal noun	今天
OD	ordinal number	第一
ON	onomatopoeia	哈哈, 哗哗
P	preposition excl. 被 and 把	从
PN	pronoun	他
PU	punctuation	、？。
SB	被 in short bei-const	被, 给
SP	sentence-final particle	吗
VA	predicative adjective	红
VC	是	是
VE	有 as the main verb	有
VV	other verb	走

Figure 4.2: The Chinese PENN Treebank tag set [30]

POS tag set of the Penn Chinese Treebank tag set and the Penn English Treebank tag set are displayed in Figure 4.1 and 4.2.

With the language island approach, the language of a word and the tagger that tags the word may differ. Therefore, we consider it as important that any word may get a suitable POS tag regardless of its language and the language of the POS tagger, which tagged the word. Consequently, we unified the given Penn English Treebank tag set and the Penn Chinese Treebank tag set. The unified set was derived based upon:

- A comparison of the Chinese Treebank tag set and the Penn English Treebank tag set from the Penn Chinese Treebank authors [30].
- A unified Chinese English Penn Treebank tag set shown in Burgmer's Diploma thesis [24].

Our unified tag set is displayed in Table 4.1. It shows the alignment of different English and Chinese POS tags to unified POS tags. For example if we consider the POS tags "adverb" in our unified set: Any POS tag, which is one of the English POS tags "RB" "RBR" "RBS" and "WRB" or the Chinese POS tag "AD", will be mapped to a single POS tag, which tags the corresponding word as an adverb. Our unified tag set differs from Burgmer's unified tag set in a way that we do not use a designated tag for English modal verbs, but align it with other verbs, since it is done so by the Chinese PENN Treebank authors [30]. Furthermore we added tags, which appeared in our data, but were missing in Burgmer's unified tag set [24]. These were the tags "OD", "AS", "ETC", "MSP", "RP", "LC", "ON", "SYM", "LS", "EX", and "POS".

POS	**English POS Tags**	**Chinese POS Tags**
Verb	VB VBD VBG VBN VBP VBZ MD	VA VE VC VV
Noun	NN NNS NNP NNPS	NN NR NT
Adjective	JJ JJR JJS	JJ
Adverb	RB RBR RBS WRB	AD
Pronoun	PRP PRP$ PP$ WP WP$	PN
Conjunction	CC	CC
Number	CD	CD OD
Determiner	DT WDT PDT	DT
Preposition	IN TO	P BA LB SB CS
Interjection	UH	IJ
De and zh particle	-	DEC DEG DER DEV
Final particle	-	SP
Measure word	-	M
Other Ch. particle	-	AS ETC MSP
Other En. particle	RP	-
Localizer	-	LC
Sound word	-	ON
Punctuation	: ; .	PU
Unknown	UNK FW	UNK FW
Symbol	SYM LS	-
Existential there	EX	-
Possessive ending	POS	-

Table 4.1: Our unified tag set based upon [30] and [24].

4.2 Text Based Prediction of LID and Code-Switch Points

Text based of language identification (LID) in a text containing CS, aims to predict the language of the current word based upon features associated to the past context. We also predict the code-switch points itself. Our work on text based prediction was mainly done at the Hong Kong University of Science and Technology.

4.2.1 Motivation of our Prediction Experiments

The motivation of our prediction experiments is to investigate the use of our applied features for multilingual language modeling with utterances containing intra-sentential CS. The challenge of recognizing speech with CS are the switches of the language. In particular the statistical estimation of n-grams in utterances containing CS is poor for common, word-based n-gram language models. We examine if we can benefit from adding language identification and POS tags as information sources.

4.2.2 Related Work

Solorio and Liu [31] proposed an approach to predict code-switch points based on Spanish-English transcriptions. They also published a paper about part-of-speech tagging of code-switched text [32].

Solorio's and Liu's paper about POS tagging inspired us to predict the language of words text based, since we will later predict code-switch points based upon the prediction of the language of words. The authors work on English-Spanish CS data of 39 minutes length. They achieve the best result applying a machine learning approach with features derived from English and Spanish POS taggers. The authors used various classifier of the WEKA [23] data mining library. They achieve the best result with the support vector machine (SVM) classifier of WEKA where they reached a mean POS tagging accuracy of 93.48%.

The paper about predicting code-switch points [31] is actually the foundation of our text based prediction of code-switch points. Solorio and Liu use a set of features related to the history of words, but not the current word. They evaluated there experiments on a recorded conversation of three bilingual English-Spanish speakers of 39 minutes length. The authors applied the Naive Bayes and Value Feature Interval classifier of the WEKA [23] data mining software. The best result they achieved for the prediction of code-switch points is an F-measure of 0.28 with a precision of 0.19 and a recall of 0.53 applying the Naive Bayes classifier. This result is not a high performance. The authors point out that it would be unrealistic to achieve a high performance for this task. For example at a position, where the machine learning approach predicts a code-switch point and the reference speaker made the decision not to code-switch, it is not implied that this particular position is unsuitable as a code-switch point. Likewise, if the classifier overlooks a code-switch point in the reference, the result may be grammatically and naturally sound. Therefore, Solorio and Liu used human evaluators to evaluate automatically generated CS sentences. On a score of 1 (worst) to 5 (best) human evaluators rated human generated CS with an average score of 3.64, while CS sentences generated with the naive bayes classifier were rated with an average score of 3.33. Randomly generated CS sentences were

rated with an average score of 2.68. The authors claim their improvement over randomly generated CS sentences to be significant according to a paired t-test. Solorio and Liu suggest the prediction of code-switch points to be used for multilingual language models, which we actually do in our factored language model experiments.

4.2.3 Predicting Language

As described earlier our goal for this section is to investigate the use of additional features for multilingual language modeling for utterances containing CS. We investigate POS tags and the language of a word as features. For predicting the language of words, we apply the following features:

- POS tags of up to six preceding words in the current utterance.
- Language (identification) of up to six previous words in the current utterance.
- The number of previous words in the current utterance.

We derived all features from our training text data. In case there are less than six words before the current word in the utterance some feature have missing values. The number of preceding words in the utterance is used, as it is the only available feature for the starting words of any utterance. For our features, it was important that all features can be generated automatically without the need of a manual human input to allow a run in a larger framework. We obtain POS tag features applying the Stanford Log-linear Part-Of-Speech Taggers [27][26] for Mandarin and English and map them to our unified tag set, described in Section 4.1. The language of individual words was obtained rule-based, by considering the Unicode number of characters of each word. The number of preceding words in the current utterance was created with a counter. The POS tag may be incorrect, since POS taggers may make errors, while the rule-based language identification and the number of preceding words should always be correct.

It is important to point out that we did not use any features derived from the current word for several reason:

- The current word along with a limited history is normally used in standard n-gram language models. We intend to investigate the use of additional features.
- Since we are working on Chinese and English which have different scripts, the language of a word can easily be obtained rule-based without error. Using the (rule-based) language identification of the current word as a feature, would be contrary to our motivation for this section. We would not gain any information about factors associated to past words, which we need for designing multilingual language models.

Given our set of 13 LID, POS and number of preceding words features, we utilized machine learning and WEKA [23] to obtain the predicted language of our current word and compare this class to its actual class value. We selected classifiers based upon popularity, variety and performance from WEKA [23]:

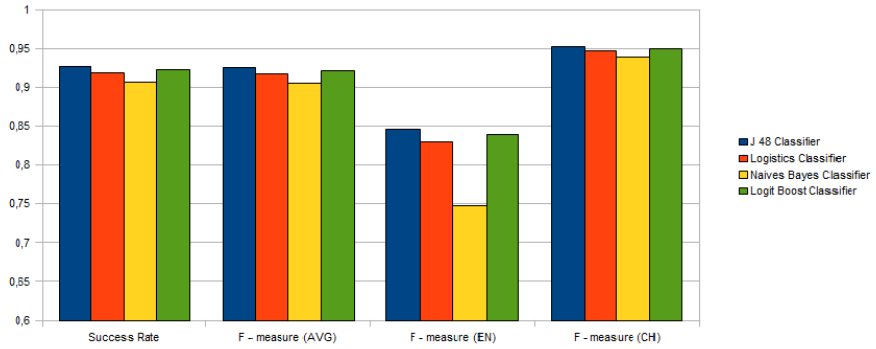

Figure 4.3: Language prediction results on our development set.

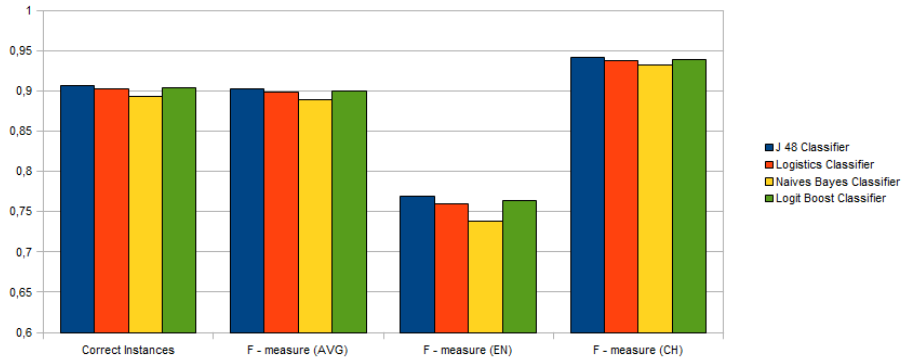

Figure 4.4: Language prediction results on our evaluation set.

- **J48:** J48 is an modified version of R. Quinlan's C4.5 [33] decision tree classifier. We activated reduced error pruning for our experiments.
- **Logistics:** The (linear) logistics regression classifier [34] creates and applies a multinomial logistic regression model with a ridge estimator.
- **Logit Boost:** Logit Boost [35] is a classifier for additive logistic regression. A regression scheme is utilized as the base learner.
- **Naive Bayes:** The Naive Bayes [36] classifier.

We investigated the use of WEKA's support vector machine (SVM) classifier (LIBSVM) [37]. However, this classifier does not support missing class values, and replacing the missing value with a default *NULL* value had a negative effect on the performance of other classifier. We used WEKA's standard value for for missing classes "?". Further the J48 classifier and the Logistics classifier performed better than the SVM classifier even with applying a default *NULL* value.

Classifier	Set	F-Measure	Precision	Recall	Correct Instances
J48	Dev.	**0.926**	**0.929**	**0.928**	**92.8%**
Logistics	Dev.	0.918	0.920	0.920	92.0%
Naive Bayes	Dev.	0.906	0.907	0.908	90.8%
Logit Boost	Dev.	0.922	0.923	0.924	92.4%
J48	Eval.	**0.903**	**0.906**	**0.908**	**90.8%**
Logistics	Eval.	0.899	0.901	0.903	90.3%
Naive Bayes	Eval.	0.890	0.890	0.894	89.4%
Logit Boost	Eval.	0.901	0.902	0.905	90.5%

Table 4.2: Results of predicted language using LID, POS tags and the count of previous words as features. The results are the weighted average of both classes.

Classifier	Set	F CH	F EN	Pr CH	Pr EN	Rec CH	Rec EN
J48	Dev.	**0.953**	**0.847**	0.926	**0.936**	**0.982**	0.773
Logistics	Dev.	0.948	0.831	0.923	0.913	0.975	0.763
Naive Bayes	Dev.	0.940	0.807	0.917	0.876	0.963	0.748
Logit Boost	Dev.	0.950	0.840	**0.927**	0.912	0.974	**0.778**
J48	Eval.	**0.942**	**0.769**	**0.915**	**0.875**	**0.972**	0.686
Logistics	Eval.	0.939	0.760	0.913	0.857	0.967	0.683
Naive Bayes	Eval.	0.933	0.739	0.909	0.823	0.958	0.670
Logit Boost	Eval.	0.940	0.764	**0.915**	0.858	0.967	**0.690**

Table 4.3: Results of predicted language for Mandarin and English. We display the F-measure (F), precision (Pr), and recall (Rec).

4.2.4 Evaluation of Predicted Language Identification

We utilized WEKA to evaluate our prediction of language with the features we described in the previous subsection. We evaluate our results with the correct instance, F-Measure, precision and recall. Table 4.2 and figures 4.3 and 4.4 show the correct instances, weighted average F-measure, precision and recall of our predicted language identification. Calculating the weighted average for both classes, recall and success rate are equal. The J48 classifier performs best for this experiment and we can achieve an F-measure of 0.926 on the development set and of 0.903 on the evaluation set. Utilizing the J48 classifier, 92.8% of our instances on our development set were classified correctly, while 90.8% of our instances on the evaluation set were classified correctly. We have two possible classes, namely the languages English and Mandarin. Mandarin is our matrix language and 74.3% of our instances on the development set and 77.6% of our instances on the evaluation set belong to the Mandarin class. Compared to a baseline where we classify every instance as Mandarin we consider our improvements to 92.8% and 90.8% to be significant.

We investigated precision, recall and the corresponding F-measure not only for the weighted average which is shown in Table 4.2 but also for the individual classes Mandarin and English which we display in Table 4.3. The language specific results show that the performance of the matrix language Mandarin is better than our results on the embedded language English. The difference in the performance is particularly in the recall, and due to its calculation also in the F-measure. With the best performing classifier regarding the recall for English which is the Logit

Boost classifier, we achieve a recall of 0.778 on the development set and 0.690 on the evaluation set. While this is not a poor result, there is a relevant difference to the average recall result. These difference indicates that our predicted language identification rather tends to predict actually English words as Chinese than any other error. Given that Mandarin is our matrix language, it is reasonable that it performs better than English. However, it is interesting that our precision for English is higher than our recall, while the weighted average (AVG) precision and recall are very close. It should be considered that the baseline, predicting every word in the Mandarin matrix language, can only result in a recall of 0.000 for the English class. Due to the fact that, for our data, the majority of words are in Mandarin, the difference between the English recall and the Mandarin recall is reasonable.

4.2.5 Features for Predicting the Language of a Word

In our previous experiment for predicting the language, we apply the actual language of previous words as features (among others). It may be argued that applying the actual language of past words as a feature for predicting the language of the current word, may be problematic. Following this argument the predicted language of preceding words should be used and not the actual one. It is reasonable that predicted language features can be omitted, because the predicted classification is exclusively based on features, which we have at our disposal. These features are the POS tags and the count of preceding words in the current utterance. If we intend to predict the language of the preceding six words and the current word, these language features would be derived from the POS tags, associated with twelve preceding words and seven features indicating the number of preceding words within the current utterances. If we have the number of previous words feature corresponding to the current word, the previous word features corresponding to past words can be derived by decrements. Furthermore it is reasonable to assume that POS tag features, corresponding to distant words, have only limited relevance for the current worst. Therefore we should only lose minor information in omitting predicted language features, as the additional features, we would apply to predict the language of preceding words, are only of limited relevance. Additionally, applying predicted language features of preceding words, would increase runtime by making it necessary to perform multiple classification runs. For these reasons, we estimate prediction experiments with omitted language features corresponding to previous words, as an adequate replacement to using predicted language features associated to previous words. For our work we consider the argument of applying predicted language features instead of actual language features, only of limited relevance. The motivation for the prediction experiments is to investigate the use of different features to be applied in multilingual language modeling for utterances with intra-sentential CS. Therefore, we are interested in obtaining information about the use of LID and POS tags as features. Performing prediction of language with exclusively utilizing POS tags of previous words and the count of previous words in the current utterance as features may help us evaluating the use of language and POS tag features. Therefore, we conduct our experiment of predicting the language of a word described in the previous section also with omitted language features of preceding words.

4.2.6 Predicting Language Identification with Omitted Language Identification Features

We performed our experiment for predicting the language of a word with omitted language features as described in the previous subsection. Except from omitting the language features to predict the language of a word, the experimental setup remains unchanged. Table 4.4 shows the result of this experiment in regards to F-measure, precision, recall and correct instances. It can be observed that the percentage of our correct instances drops from 92.8% on the development set and 90.8% on the evaluation set to 79.7% and 81.7% respectively always considering the best performing classifier. Note that the fraction of Chinese words is 74.3% on the development set and 77.6% on the evalution set. Also the F-measure decreases from 0.93 on the development set and 0.90 on the evaluations set to 0.76 and 0.78, respectively. We also evaluate the class specific results regarding precision, recall and F-measure, shown in Table 4.5. It can be derived that in addition to the overall performance loss, the recall on English words seems to be rather low comparing to the other measures. 69.5% of our English words are predicted as Chinese on our development set and 71.9% on the evaluation set, respectively, considering the best performing recognizer for the English recall. We achieve a better performance on the evaluation set than on the development set. This may be connected to the fact that the share of words in our matrix language (Mandarin) is higher on the evaluation set than on our development set. However, in our experiments with utilizing language features, we achieved a better performance on the development set than on the evaluation set. Comparing the classifiers, we realize that unlike in our experiments, where we included the actual language of preceeding words as a feature where the J48 classifier [33] performs best, the Logistics classifier [34] and the Naive Bayes classifier [36] provide the best performance. Our goal is not to compare different classifiers, but to apply classifiers which provide reasonable results. Our key observations of this experiment without utilizing the actual language are the following:

- Omitting the language identification of preceding words from the features results in a significant performance loss in particular to the recall of English words.
- Compared to the baseline, where we consider every word as Chinese, we still achieve an improvement.

From this observations, we conclude that the actual (accurate) language identification of preceding words in the current utterance is an important feature for predicting the language identification of the current word. Note that due to our unified POS tags, it is in general not possible to obtain the language identification of a word from its POS tag, unless the POS tag belongs to a class, where only tags from one language were mapped to. POS tags in every class of Table 4.1, where either no English or no Chinese POS tags were associated to, directly indicate the language of its corresponding word. For example words tagged as a measure word can only be Chinese, while for example nouns can belong to both languages.

Figures 4.5 and 4.6 show the performance of our language identification experiments on development set and evaluation set. We compare our results with a baseline

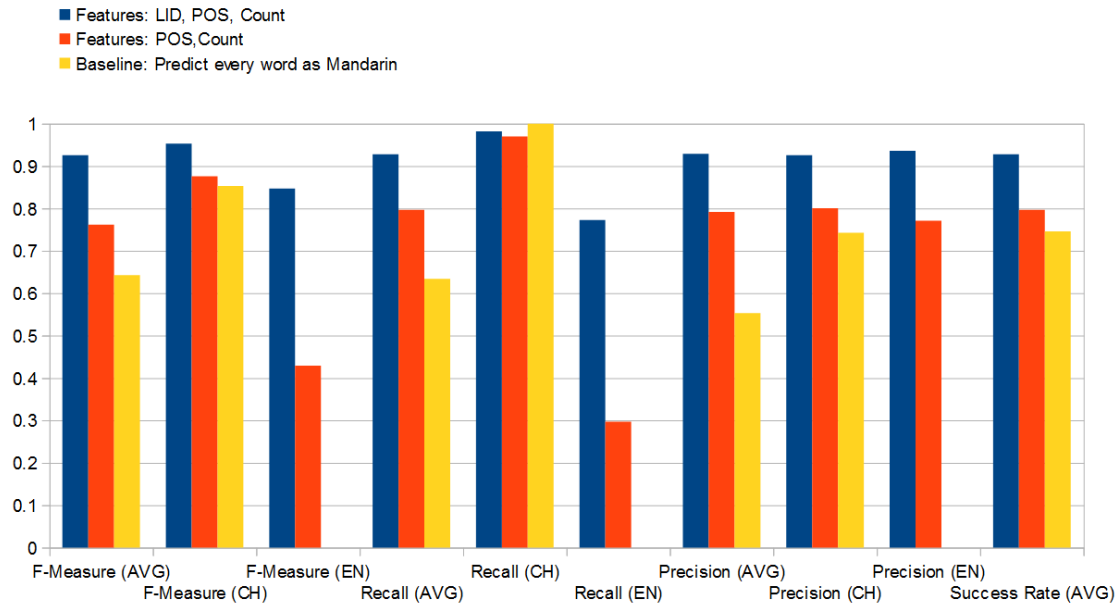

Figure 4.5: F-measure, precision and recall of our language identification experiments on our development set.

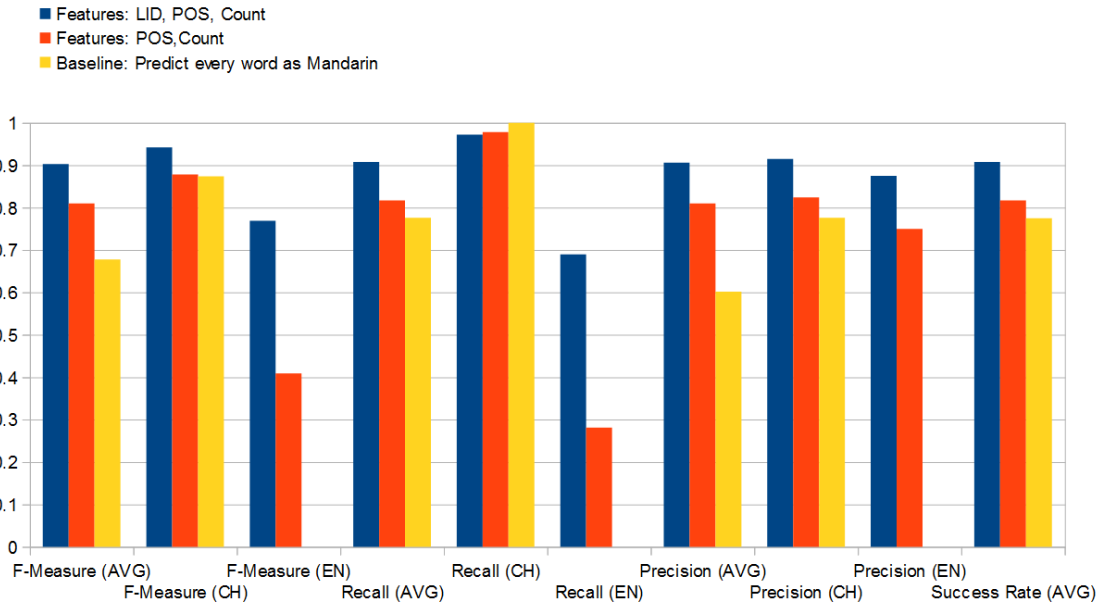

Figure 4.6: F-measure, precision and recall of our language identification experiments on our evaluation set.

Classifier	Set	F-Measure	Precision	Recall	Correct Instances
J48	Dev.	0.715	0.750	0.768	76.8%
Logistics	Dev.	**0.762**	**0.792**	**0.797**	**79.7%**
Naive Bayes	Dev.	0.761	0.787	0.795	79.5%
Logit Boost	Dev.	0.696	0.733	0.759	75.9%
J48	Eval.	0.743	0.762	0.791	79.1%
Logistics	Eval.	0.779	**0.810**	**0.817**	**81.7%**
Naive Bayes	Eval.	**0.784**	0.807	**0.817**	**81.7%**
Logit Boost	Eval.	0.743	0.768	0.793	79.3%

Table 4.4: Results of predicted language identification using POS tags and the count of previous words as features. The results are the weighted average of both classes.

Classifier	Set	F CH	F EN	Pr CH	Pr EN	Rec CH	Rec EN
J48	Dev.	0.862	0.292	0.775	0.679	0.970	0.186
Logistics	Dev.	**0.876**	0.429	0.800	**0.771**	0.970	0.297
Naive Bayes	Dev.	0.875	**0.433**	**0.801**	0.747	0.964	**0.305**
Logit Boost	Dev.	0.857	0.231	0.766	0.636	**0.972**	0.141
J48	Eval.	0.878	0.275	0.803	0.621	0.969	0.176
Logistics	Eval.	**0.892**	0.387	0.820	**0.775**	**0.978**	0.258
Naive Bayes	Eval.	**0.892**	**0.409**	**0.824**	0.748	0.973	**0.281**
Logit Boost	Eval.	0.879	0.273	0.803	0.646	0.973	0.173

Table 4.5: Results of predicted language identification for Mandarin and English. We display the F-measure (F), precision (Pr), and recall (Rec).

where we predict every word as Chinese. With both our experiments, we achieve a significant improvement considering the weighted average for F-measure, precision and recall.

4.2.7 From Predicting Language Identification to Predicting Code-Switch Points

As we work with English-Mandarin data, containing intra-sentential CS, predicting code-switch points is particularly useful. Since our data is in Mandarin and English the possible code-switch points can either be from Mandarin to English or from English to Mandarin. For this experiment, we do not differentiate between these two types of code-switches since both are code-switch points. Furthermore, we exclusively consider intra-sentential code-switches because they are the most challenging. In case one utterance ends in Chinese or English, while the following utterance start in the other language we ignore this inter-sentential code-switch. The prediction of the language of a word and the predicting of code switching points is related. To derive the prediction of code-switch points from prediction of language identification we need to compare the predicted language and the actual language of neighboring

Act LID w_0	**Act LID w_1**	**Pred LID w_0**	**Pred LID w_1**	**Evaluation**
L1	L2	L1	L2	true positive
L1	L2	L2	L1	true positive
L1	L1	L1	L1	true negative
L1	L1	L2	L2	true negative
L1	L1	L1	L2	false positive
L1	L1	L2	L1	false positive
L1	L2	L1	L1	false negative
L1	L2	L2	L2	false negative

Table 4.6: This table shows how code-switch points can be derived from the LID. The predicted (Pred) and actual (Act) LID are considered.

words. If we would be able to predict every language correctly, we would also be able to predict every code-switch point correctly. Knowing the language of two consecutive words is sufficient to determine if a code-switch point is happening. We define the words, which we observe as w_0 and w_1 where w_1 is the succeeding word of w_0 in the same utterance. L1 and L2 individually stand for one of our languages in our data, in particular Chinese or English. L1 and L2 can be either of these languages, but never the same. The possible cases are shown in table 4.6. Note that we do not evaluate, whether the predicted language identification for w_0 and w_1 is correct, but whether the derived predictions and the reference are code-switch points.

4.2.8 Evaluation of Predicted Code-Switch Points

We evaluated our predicted language against the code-switch points in the reference utilizing the same classifiers as in our previous experiments. Due to the better results, we derive our predicted code-switch points from our predicted language identification, where we included the actual language identification of past words as features. There is a code-switch in 6.56% of our instances on the development set and 8.25% of our instances on the evaluation set. Therefore, true negatives (positions, where we have predicted to be no code-switch and there actually is none) are not that interesting to evaluate for this experiment. Consequently, we focus on precision, recall and F-measure for evaluation. The results are displayed in Table 4.7. Figure 4.7 shows an overview of the F-measure results, considering the results of our different classifiers. The logistics classifier [34] performs best. Please note that due to a high number of true negatives the correct instances are about 90% for all applied classifiers on both sets. However, an F-measure of 0.205 on the development set and 0.300 on the evaluation set is rather low. In order to compare the results to the baseline we applied for predicting language identification is not feasible as it classified every word as Chinese and predicts 0 code-switch points.

We considered two random baseline approaches for predicting code-switch points and compare them to our result. The first approach has a probability of 50% to predict an instance as a code-switch point, due to two possible classes (code-switch point, no code-switch point). In the second approach the probably to predict an instance as a code-switch point is equal to the fraction of code-switch points in

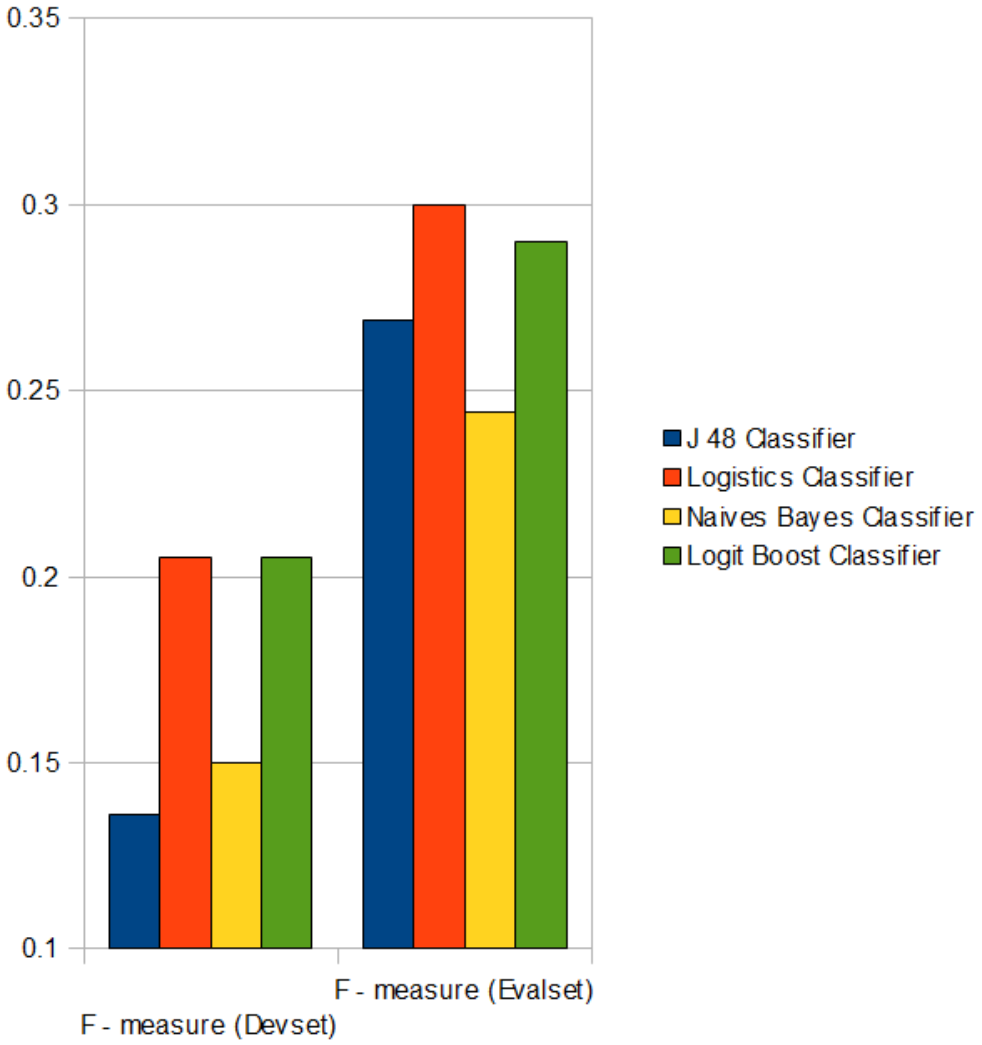

Figure 4.7: F-measures of our CSP prediction experiment.

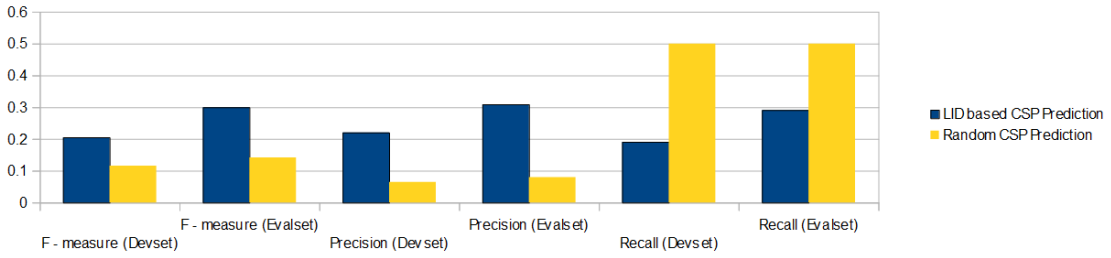

Figure 4.8: Results of our experiment to predicting code-switch points for development set (Devset) and evaluation set (Evalset) compared to random predictions.

Classifier	Set	F-Measure	Precision	Recall
J48	Dev.	0.136	0.162	0.117
Logistics	Dev.	**0.205**	**0.220**	**0.191**
Naive Bayes	Dev.	0.150	0.182	0.123
Logit Boost	Dev.	**0.205**	**0.220**	**0.191**
J48	Eval.	0.269	0.301	0.243
Logistics	Eval.	**0.300**	0.309	**0.291**
Naive Bayes	Eval.	0.244	**0.328**	0.194
Logit Boost	Eval.	0.290	0.311	0.272

Table 4.7: Results of predicted code-switch points derived from our predicted language identification.

the set. For these baselines, we assume the recall to be equal to the fraction of predicted code-switch points in order to simulate a random prediction. Table 4.8 compares our prediction of code-switch points with the random prediction of code-switch points, which has a 50% chance to predict code-switch points and a recall of 50%. Calculating the first random approach, we achieve an F-measure of 0.116 on the development set and 0.142 on the evaluation set for the first experiment. For the second experiment, we achive an F-measure of 0.033 on the development set and 0.041 on the evaluation set. Both baseline results are significantly worse than our best F-measure of 0.205 on the development set and 0.300 on the evaluation set. For this experiment, a high F-measure is not realistic because, as described in [31], the speaker has a high flexibility in the decision where to code-switch. Even if we have suitable conditions for a code-switch point, the reference speaker may still decide not to code-switch. The reference is only one realization of the speaker's choice. That means our result evaluated on the baseline shows, what we can achieve at least. In [31], CS sentences are artificially generated and evaluated by human evaluators. We consider this labor intensive task as unnecessary for our work, because our results indicate that utilizing POS tags, language identification, and the count of preceding words provides useful information for predicting code-switch points. Furthermore, we are proceeding to the next step, where we do multilingual language modeling on utterances containing CS.

4.2.9 Summary of our Prediction Experiments

From our experiments of predicting the language of a word and code-switch points, we conclude the following results:

- Based upon language identification from text, POS tags and the count of preceding words in the current utterance, we can predict the language of the current word with a success rate of 92.8% on the development set and 90.8% on the evaluation set, respectively. The baseline provides a success rate of 74.3% on the development set and 77.6% on the evaluation set.
- Omitting the actual language identification from the features leads to a performance drop to 79.7% on the development set and 81.7% on the evaluation set.

Set	PPL 2-gram	PPL 3-gram	PPL 4-gram	OOV
Development	342.1	333.2	332.0	7.76%
Evaluation	345.4	338.6	337.9	8.14%

Table 4.8: OOV and perplexity(PPL) of 2-gram, 3-gram, and 4-gram language models, built on our training data.

- For predicting code-switch points based upon language identification, POS tags and the count of preceding words in the current utterance, we achieve an F-measure of 0.205 on the development set and 0.300 on the evaluation set. Compared to the baseline where we can achieve an F-measure of 0.116 on the development set and 0.142 on the evaluation set, our results are still an improvement.

Since we achieved an improvement in our experiments on predicting code-switch points, we conclude that the language of a word and POS tags in combination contain information about code-switch points. In our experiments on predicting the language of a word, we observed that applying POS tags of preceding words and the number of preceding words, provides a performance improvement compared to our baseline, where we predict every word as Chinese. However we get a more significant improvement, if we include the actual language of preceding words as features. Since we derived our prediction of code-switch points from our prediction of language, also the individual factors may contain information for predicting code-switch points. Therefore we apply language and POS tag features individually and in combination in our factored language model experiments described in Section 4.5.

4.3 Baseline Speech Recognition System

The amount of training speech data in our database is not sufficient to build a speech recognizer with good performance. A sufficiently large multilingual Mandarin-English speech database was not available in the begin of this work. Since Mandarin is our matrix language, we apply a Mandarin speech recognizer and perform an adaption using our training speech data. We utilize the CMU-InterACT 2008 Mandarin Transcription System [38] as Mandarin speech recognizer. In particular, we use the AM1 system, which is described as a speaker independent initial-final model applying multi-style training utilizing 1,300 hours broadcast news and broadcast conversation data for training. The AM1 system used maximum likelihood (ML) training with about 6,000 codebooks. We perform speaker-adaptive training (SAT) with feature space adaptation (FSA) on the AM1 system utilizing our training set of our Mandarin-English speech corpus. We investigated the use of 2-gram, 3-gram, and 4-gram language models built exclusively on our 14983 word tokens of training data. The 2338 different words of the training data were used as our vocabulary. Table 4.8 shows an overview about the perplexities and OOV rates. The OOV rate is independent of the n-gram order. It can be observed that the 4-gram language models has the lowest perplexity on the development and evaluation set. The 3-gram language model has a marginally higher perplexity than the 4-gram language model on

both set. The 2-gram language model has the highest perplexity among the three language models on both sets. Due to the perplexity results, we apply the 4-gram language model built upon our Mandarin-English CS training set for the decoding process. Considering our limited text data and the marginal perplexity difference between the 3-gram and the 4-gram language model, we investigated the use of a 3-gram language model instead of a 4-gram. On the development set the (rounded) MER was equal applying the 3-gram or the corresponding 4-gram language model. However, the number of errors on the system with the 4-gram language model was marginally lower. We built the language model with the SRI language model toolkit [15] using the original (unmodified) Kneser-Ney smoothing [39].
Our baseline system achieved a MER of 59.1% on our development set and 60.8% on our evaluation set. Note that these error rates are rather high, but it has to be considered that we recognize conversational and meeting data, which is more difficult to recognize than read speech. Also intra-sentential CS proposes a challenge for automatic speech recognition, which affects the error rate compared to monolingual speech data.

4.4 Factored Language Models - Fundamentals

This section describes the fundamentals of factored language models (FLM) [3]. The idea of this section is to describe what factored language models are and to point out the challenges for factored language models. Furthermore, the FLM specification is shown. This section is based upon the FLM Tutorial by Kirchhoff et al. [40].

FLMs have been introduced by Bilmes and Kirchhoff to incorporate morphological information into language modeling. They are a flexible framework for including information sources, like part-of-speech, into language modeling. In FLMs, a word w_t can be seen as a set of K factors:

$$w_t = \{f_t^1, f_t^2, \cdots, f_t^K\} \tag{4.1}$$

The factors itself can be anything connected to the current word. Examples for factors are part of speech, morphological classes, word stems or even the word itself. Using the word itself as a factor is particularly useful since the probabilistic language model can be built upon the words and additional factors. The statistical language model is estimated over the set of overall factors:

$$p(f_{1:T}^{1:K}) \tag{4.2}$$

FLMs provide many different modeling options in addition to those for standard n-gram language models. With the chain rule of probability we can get:

$$\prod_k p(f_t^k | f_t^{1:k-1}, f_{t-n+1:t-1}^{1:K}) \tag{4.3}$$

This is only one possibility of chain rule ordering. There can be 2^{nK} possible options for conditioning $p(f_t^k | \cdots)$.
The key design challenges for creating an FLM are:

1. Choosing a suitable set of factors. Data-driven techniques or linguistic knowledge can be used for this task.

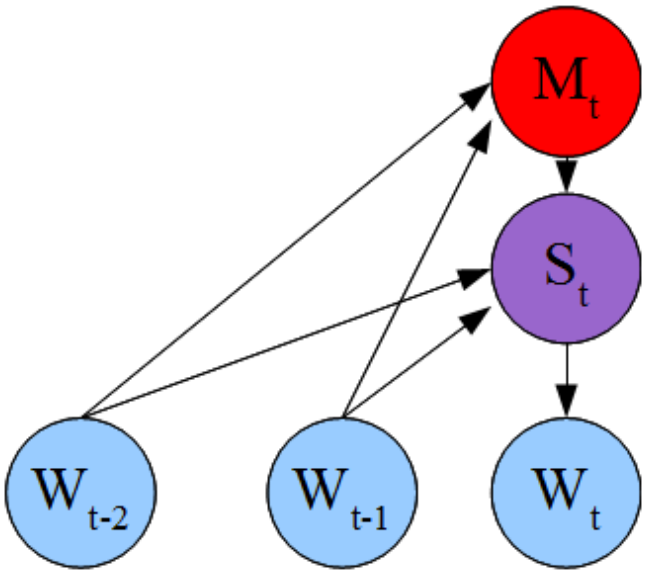

Figure 4.9: Example FLM over words W_t, morphological factors W_t and stems S_t [40].

2. Determine the best statistical model of the chosen factors. The goal is to limit the estimation difficulties, which are present in standard n-gram models and to obtain accurate predictions for the statistical properties of the applied languages.

Figure 4.9 corresponds to the following model:

$$p(w_t|s_t, m_t)p(s_t|m_t, w_{t-1}, w_{t-2})p(m_t|w_{t-1}, w_{t-2}) \tag{4.4}$$

In the FLM of this example the word is dependent of its stem and the morphological class. If the assumption of this dependency is correct the perplexity of $p(w_t|s_t, m_t)$ should be low (close to unity). The goal of this model is that the product of the perplexities $p(w_t|s_t, m_t)$, $p(s_t|m_t, w_{t-1}, w_{t-2})$ and $p(m_t|w_{t-1}, w_{t-2})$ is lower than the perplexity of a traditional trigram. From an information-theoretic perspective, the language model that minimize the perplexity is closest to the true probability distribution of the language [2].
It may be helpful adding M_{t-1} and S_{t-1} as additional parents of M_t to the FLM because they might decrease the perplexity of $p(m_t|\cdots)$. However, adding parents increase the difficulty of the estimation problem, i.e. the dimensionality. Therefore, the goal for obtaining suitable factors for an FLM is to find the best balance between predictability and estimation error. The predictability influences the entropy and perplexity of the language model and should increase with additional parents as long as they provide useful information. Finding the best balance is a common problem in the field of statistical inference, where variance and bias may be traded for each other [40].
Another important point for designing the structure of an FLM is to consider that some factors may be unknown. Given the example

$$p(w_t|w_{t-1}, w_{t-2}, s_{t-1}, s_{t-2}, m_{t-1}, m_{t-2}) \tag{4.5}$$

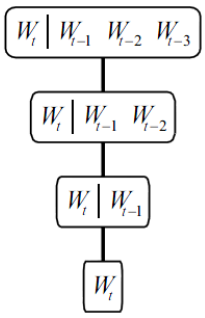

Figure 4.10: Example backoff path of a 4-gram language model [40].

Compared to the previous example, previous stems s and morphs m have been included as parents. Normally, stems and morphs are deterministic functions of the corresponding word. That means that we cannot usually gain additional information from the stem and the morph if we know the word itself. Still, it may be useful to include the stems and morphs as a parent as there may be n-grams which do not occur in the training data, but the word and the preceding stems and morphs do. In such a case, we would gain information by backing off to the word and preceding stems and morphs compared to just the word itself.

4.4.1 Backoff for FLMs

For standard n-gram language models, it is reasonable to drop the most distant parent first, linearly continuing with the parents, which are closer to the word. For FLMs, it is a more difficult problem since there may be many different factors in the language model.
Backoff is applied if there is insufficient training data available to fully estimate a high-order conditional probability table [40]. Therefore, only a portion of the table is estimated while the remainder is calculated from a lower-order model by dropping at least one of the variables from the conditions, for example from a 4-gram $p(w_t|w_{t-1}, w_{t-2}, w_{t-3})$ to a 3-gram $p(w_t|w_{t-1}, w_{t-2})$.

Figure 4.10 shows the standard backoff path of a 4-gram language model. As long as the string $w_{t-3}\ w_{t-2}\ w_{t-1}\ w_t$ appears with sufficient frequency in the training data, the 4-gram $p(w_t|w_{t-1}, w_{t-2}, w_{t-3})$ is estimated. Otherwise a backoff to a lower order n-gram is applied, where most distant words, for example w_{t-3}, are dropped. The procedure can be seen as a graph. Example backoff graphs are shown in figure 4.11.

For FLMs, we need distributions of the form $p(f_t^i|f_{t_1}^{j_1}, f_{t_2}^{j_2}, \cdots, f_{t_N}^{j_N})$ with $k = 1...N$, a conditional probability table in the most general case. For simpler notation, we write the model in the following form:

$$p(f|f_1, f_2, ..., f_N) \tag{4.6}$$

F is the child variable with N parent variables $F_1...F_N$ and f is the possible value of F. If we only have words as factors in the FLM, it is common to drop the most distant word first. However, if there are factors involved which are not exclusively words, it is not obvious, which factors are to drop in which order. Even if assumed that only one parent is dropped at a time, there is a very large number of possibilities.

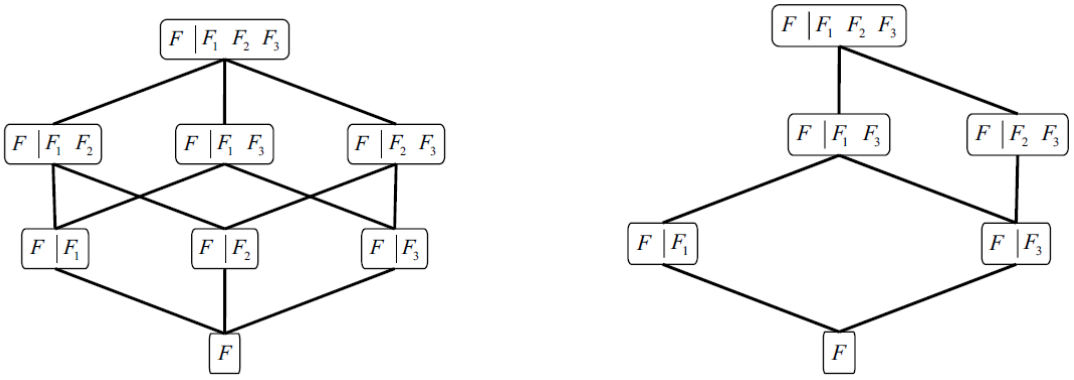

Figure 4.11: Backoff graphs for $p(F|F_1, F_2, F_3)$ showing all possible backoff paths (left), or where only a subset of possible backoff paths is allowed (right) [40].

Furthermore, it is possible to drop more than one parent or add parents in a backoff step.

There are several approaches for choosing the backoff paths [40]. Among those are:

- Dropping always the most distant parent, as it is done in typical word based n-gram language models.
- Dropping the parent, which is found to contain the least information in an information-theoretic sense.
- Dropping the parent based on a tradeoff between statistical predictability and estimation.
- Generalized all-child backoff: Multiple child nodes in the backoff graph are chosen during run-time. This will be described in the next subsection.
- Generalized constrained-child backoff: A subset of the child nodes in the backoff are chosen at run time. This will also be described in the next subsection.

4.4.2 Generalized Backoff

Generalized backoff [40] allows to choose multiple dynamical paths at runtime, when moving from a higher level to a lower level in a backoff graph instead of choosing a fixed path. For generalized backoff there are two options:

- Only one path is chosen at a time, but dependent on the particular sequence of words or factors which need a probability to be estimated, the backoff path may change.
- Multiple backoff paths are applied simultaneously at runtime. Precisely, multiple paths will be used to generate a probability. The set of multiple paths applied depend on the particular instance of the factor sequence.

Generalized backoff can improve over choosing only a fixed backoff path, since the applied paths can be selected to suit best for the given factor instances. In our experiments we mainly apply FLMs with only one backoff path, but we also utilize one FLM with multiple backoff paths.

4.4.3 FLM Training Data Format

Normally, training text for the language model consists of a text corpus. For an FLM each word may be attached with a set of factors. Consequently, a stream of vectors instead of a stream of words must be specified. An FLM is a model over multiple streams of data, each stream corresponding to a factor in the model [40]. Each factor receives a tag attached with dash "-" and different factors corresponding to a word are separated with ":". For words, the tag W is used and may be omitted. If the value of a factor corresponding tag is missing $NULL$ is used. The order of the factors is irrelevant [40].
Example:

```
speaking is natural
```

If we use the word itself (W) and POS tag (P) as factors, we get:

```
P-noun:W-speaking P-verb:W-is P:adjective:W-natural
```

4.4.4 FLM File

The FLM file [40] consists of at least one specification of a given FLM. Multiple FLM specifications are trained simultaneously and used for perplexity calculation or N-best rescoring. Comment lines start with #. Ignoring comment lines, the first line specifies the number of FLMs. Then a header line follows, containing the specification of the child followed by a ":", number of parents, the parents itself, a count file name, a language model file name, and the number of backoff node lines to follow. A backoff node line must not appear before the header line and consists of a set of parents, a set of parents to drop and possibly node options. The header line of FLM syntax is defined:

```
child: num_parents par_1 par_2 ... par_N count_file
lm_file num_of_backoff_nodes
```

After the header line there are node backoff lines:

```
parent_list drop_list node options
```

The parent_list is a comma separated list, displaying the parents corresponding to this node. The drop_list is a comma separated list of the parents to be dropped at this stage. The drop_list is a subset of the parent_list. The child variable is the random variable, which we want to predict. In normal n-grams and in all of our experiments with FLMs, this would be the words. The parents are specified with a time offset, which indicates the position of the parent in relation to the current node (negative offsets specify the past, positive specify the future). An example for a standard trigram language model looks as follows [40]:

```
## standard 3-gram LM using unmodified Kneser-Ney discounting
W: 2 W(-1) W(-2) word_3gram.count word_3g.lm 3
W1,W2 W2 ukndiscount gtmin 3 interpolate
W1 W1 ukndiscount gtmin 1 interpolate
0 0 ukndiscount gtmin 1
```

In this example, we drop the temporally most distant parent first. We remove W2 which is the word appearing two words before the current word. Then we delete the following word of W2, namely W1. *0* is indicating that there are no parents and no parents are dropped therefore.

Another example for a trigram language model this time with a time-reversed backoff path [40]:

```
## time reversed 3-gram LM
W: 2 W(-1) W(-2) word.count word_3g.lm 3
W1,W2 W1 ukndiscount gtmin 3 interpolate
W2 W2 ukndiscount gtmin 1 interpolate
0 0 ukndiscount gtmin 1
```

In this example we drop the word preceeding the current word (W1) before its predecessor (W2).

To display the use of generalized backoff, an example for a trigram with generalized backoff is shown [40]:

```
## 3-gram LM with generalized backoff
W: 2 W(-1) W(-2) word.count word_3g.lm 4
W1,W2 W1,W2 ukndiscount gtmin 3 interpolate
W2 W2 ukndiscount gtmin 1 interpolate
W1 W1 ukndiscount gtmin 1 interpolate
0 0 ukndiscount gtmin 1 kn-count-parent W1,W2
```

We have four backoff nodes in this example both W1 and W2 can be dropped in the first step.

It is possible to skip a level in the FLM backoff-graph by setting the minimum counts (gtmin) to a very high value that the node will not be reached [40]:

```
## word given word morph (M) stem (S)
W: 3 W(-1) M(-1) S(-1) dev.count dev.lm 5
W1,M1,S1 W1 ukndiscount gtmin 2 interpolate
M1,S1 M1,S1 ukndiscount gtmin 100000000 combine mean
M1 M1 ukndiscount gtmin 1 kn-count-parent W1,M1,S1
S1 S1 ukndiscount gtmin 1 kn-count-parent W1,M1,S1
0 0 ukndiscount gtmin 1 kn-count-parent W1,M1,S1
```

In this example, we have an FLM using words (W), morphs (M) and stems (S) as factors. First W1 is dropped. The node M1,S1 has a very large minimum count. It does not hit. The scores of this node have to be combined from lower-level nodes, i.e. from the mean of $p(W_t|M_{t-1})$ and $p(W_t|S_{t-1})$, because S1 and M1 may be dropped individually.

4.4.5 Data-driven Search of FLM Parameters

According to [40], there are different types of parameters to specify an FLM:

- Initial conditioning factors: Which factors shall be used for n-gram probability estimation?
- The backoff graph and smoothing options: How is robust estimation processed in case of insufficient data?

The goal of the data-driven parameter search is to get a low perplexity on unseen data. For a factored word representation with k factors, there are $\sum_{n=1}^{k} \binom{k}{n}$ possible initial conditioning factor subsets. Given we have m conditioning factors, up to $m!$ backoff paths are possible, each with its own smoothing options. Thus for large m, an exhaustive search is not feasible. Furthermore, these parameters interact with each other making the search especially difficult. The performance of an FLM structure is data dependent. In [41], a genetic algorithm for tuning FLMs named GA-FLM was developed. It applies the training text and a development text and attempts to find FLM parameters minimizing the perplexity on the development set.

The program flow of GA-FLM is illustrated in Figure 4.12 and works in the following way:

1. Initialize a population of genes, where each gene represents an FLM with specific conditioning factors, backoff path and smoothing options.
2. From each gene the FLM file is created and the language model is trained and tested.
3. The fitness value of the gene is determined by the perplexity of the user supplied development set.
4. Selection, crossover and mutation is applied in order to get the next population of genes.
5. GA-FLM runs until convergence or the maximum generations specified by the user is reached.
6. The best gene of each generation is included in the next generation.

In section 4.5 we build a FLM using GA-FLM.

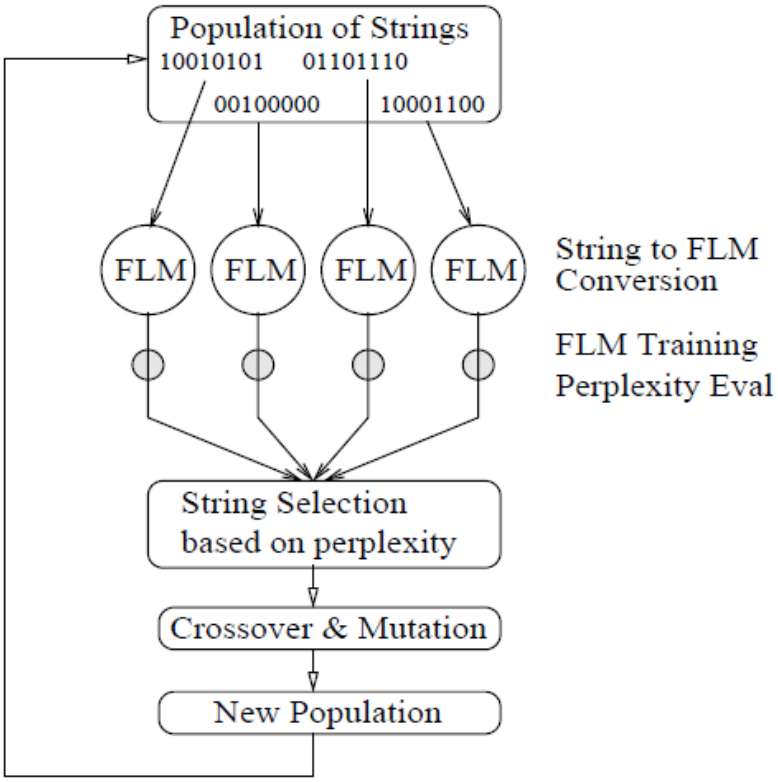

Figure 4.12: Program flow of genetic algorithm (GA) search for FLM structure [41].

4.5 FLMs for Recognizing Utterances with intra-sentential CS

This work was mainly done at the Cognitive Systems Lab at the Karlsruhe Institute of Technology. Utterances with intra-sentential CS imply a special challenge for automatic speech recognition, particularly for the language model. While monolingual utterances are comparatively easy to handle with statistical n-gram language models, utterances with CS are problematic since the language model needs to model a probability for language switches implied in the common probability of word sequences. Therefore, it is reasonable to use further information in the language in addition to words. The FLM [40] allows to use a set of factors in addition to normal words in the standard n-gram language model. Consequently, we decided to apply FLMs for speech recognition of speech data with intra-sentential CS. Note that to our best knowledge, factored language models have not been used to recognize speech containing intra-sentential CS before. Our approach uses the following factors:

- Common words,
- Language of each word,
- Part of speech tag of each word,
- Likelihood of a code-switch after the current word.

Utilizing words as a factor is obvious, since they are applied in common n-gram language models and are useful to estimate the probabilities of word sequences.
The language of each word is a surjective function of its corresponding word, as long as we are only considering English and Chinese. Note that there are no words shared between those languages. Consequently, if we we know a word, we do not gain information from its language, since we already are aware of the language corresponding to the known word. Therefore, the language is only helpful in case a word (which may be in the history) is an out-of-vocabulary (OOV) word. In our work a word is an OOV word, if it is not covered by the language model. The vocabulary in our specified vocabulary file matches the vocabulary in our baseline 4-gram language model, which was exclusively built on the training transcriptions. If we have a word which is OOV, its language is still known, because all possible values of the language ID (which are Chinese and English in our experiments) appear in the training data.
The utilized POS taggers are described in Section 4.1. Our prediction experiments in Section 4.2 motivate the use of POS tagging and language factors as these features are useful for predicting language and code-switch points. POS tags are context-dependent and provide syntactic information, which may be helpful to get a more suitable estimation for utterances containing intra-sentential CS. As displayed in an example of [1] described in section 2.1, CS may happen at sentence or utterance boundaries, too. However, our work focus on intra-sentential CS, since it is the more challenging and general problem.
We investigated if there are key words in the training set after which a code-switch is likely to happen. Therefore we estimated the likelihood of a code-switch after each word on the training set. Since our code-switch corpus is rather small, we made a 1-gram estimation. We split the probabilities into two, three, and five classes with an equally distributed likelihood range. It was a class added for words, which do not appear in the training data. We added the code-switch point probability classes as a factor for the FLM training data. Given we have the following phrase:

```
YOUR CHINESE IS REALLY GOOD
```

The representation for the FLM training data looks like

```
YOUR:P-PN:L-en:S-CS0 CHINESE:P-NN:L-en:S-CS0 IS:P-VV:L-en:S-CS0
REALLY:P-AD:L-en:S-CS0 GOOD:P-JJ:L-en:S-CS1
```

P indicates the POS tag, while L indicates the language ID and S the code-switch probability class. The different factors are separated by ":". In this example, we have an English utterance without any code-switch points. Therefore the language ID for each word is *en*. Furthermore, after each of these words, except *GOOD*, a code-switch is not likely. *GOOD* has the code-switch probability class CS1, indicating a following code-switch is likely. All other words in our example have the code-switch probability class CS0, which indicates that a code-switch is unlikely. The POS tagger provides appropriate tags for the words. For example, the word *YOUR* is tagged as a pronoun (PN).

4.5.1 N-Best Rescoring with Factored Language Models

The direct use of factored language models in the decoder is not supported in the Ibis decoder [25]. Therefore, we decided to use factored language models in the rescoring process. As described in Section 4.4, FLMs require their own text data format if factors are different from words. We intended to achieve a suitable integration for the FLMs, working on data in the appropriate FLM data. We generated the lattice with the standard n-gram baseline language model and created a number of hypotheses, which are to be reranked with factored language models. This reranking is called N-best rescoring. The best reranked hypothesis is evaluated for each utterance and a combination of N-best rescoring parameters.

4.5.2 FLM Design

While we have described the choice of the factors of our FLMs earlier, this subsection describes the structures of our FLMs. Most of our FLMs have been designed knowledge-based, while we created a few FLMs data-driven. For the knowledge-based FLM design, we aimed to incorporate additional information compared to our 4-gram baseline language model. Consequently, we included a history of three words in all our knowledge-based FLMs, so that we avoid losing information on word level. Additionally, we intended to be able to evaluate each additional factor (language, POS tags, code-switch point probability) individually and their combination as well. Considering that our speech data with English-Mandarin CS is rather small, the data should favor a estimation in a lower dimension over possible more information by adding additional parents. However, we investigated a varying number of parents for the LID. For all our language models, we applied the original (unmodified) Kneser-Ney discounting [39], since Kneser-Ney discounting tends to provide the best performance (for n-gram language models) compared to other discounting methods. We utilized Kneser-Ney discounting for FLMs, since we applied this discounting method for our n-gram baseline language model and to investigate different discounting methods was not our focus. We used the original (unmodified) Kneser-Ney discounting, since previous experiments have shown that modified Kneser-Ney discounting is more likely to provide problems on small data, which we have for some of our applied factors. Furthermore, comparing different discounting methods is not our objective. We created the following FLMs for rescoring:

- built on words and one LID parent (LID1).
- built on words and three LID parents (LID3).
- built on words and one unified POS tag parent (POSU).
- built on words and one POS tag parent, where the POS tags, processed with the Mandarin tagger, remain unmapped, while all POS tags, processed with the English tagger, are mapped to a special tag *EN* (POSEN).
- built on words and one code-switch probability parent with 2 classes (Prob2).
- built on words and one code-switch probability parent with 3 classes (Prob3).
- built on words and one code-switch probability parent with 5 classes (Prob5).

FLM Name	LID_{t-1}	POS_{t-1}	CSP_{t-1}	$LID_{t-2\&3}$	**Comments**
LID1	yes	no	no	no	-
LID3	yes	no	no	yes	-
POSU	no	yes	no	no	unified POS tags
POSEN	no	yes	no	no	Mandarin POS tags unmapped, all English POS tags to "EN"
Prob2	no	no	yes	no	2 classes for CSP probability
Prob3	no	no	yes	no	3 classes for CSP probability
Prob5	no	no	yes	no	5 classes for CSP probability
LIDPOS	yes	yes	no	no	unified POS tags
LIDPOSProb	yes	yes	yes	no	2 classes for CSP probability, unified POS tags
GA	yes	yes	no	no	data-driven built with GA-FLM, includes additional parents

Table 4.9: FLM designs and the applied parent factors related to word history t.

- built on words, one LID parent and one unified POS tag parent (LIDPOS).
- built on words, one LID parent, one unified POS tag parent and one code switch-probability parent with 2 classes (LIDPOSProb).
- built data-driven with GA-FLM [41] on LID, unified POS tags and code-switch probability with 2 classes (GA). The backoff nodes and paths were chosen by GA-FLM while we limited the discounting method to the original (unmodified) Kneser-Ney discounting.

Table 4.9 shows an overview about different designs for factored language models. With the following example we describe the design of the FLM labeled as LIDPOSProb in detail and describe differences to other FLMs. The FLM file corresponding to our FLM LIDPOSProb is the following:

```
1
W : 6 W(-1) W(-2) W(-3) P(-1) L(-1) S(-1) trainALL.count trainALL.lm 7
W1,W2,P1,W3,L1,S1 W3 ukndiscount gtmin 0 interpolate
W1,W2,P1,L1,S1 W2 ukndiscount gtmin 0 interpolate
W1,P1,L1,S1 W1 ukndiscount gtmin 0 interpolate
P1,L1,S1 S1 ukndiscount gtmin 0 interpolate
P1,L1 P1 ukndiscount gtmin 0 interpolate
L1 L1 ukndiscount gtmin 0 interpolate
0 0 ukndiscount gtmin 0 interpolate
```

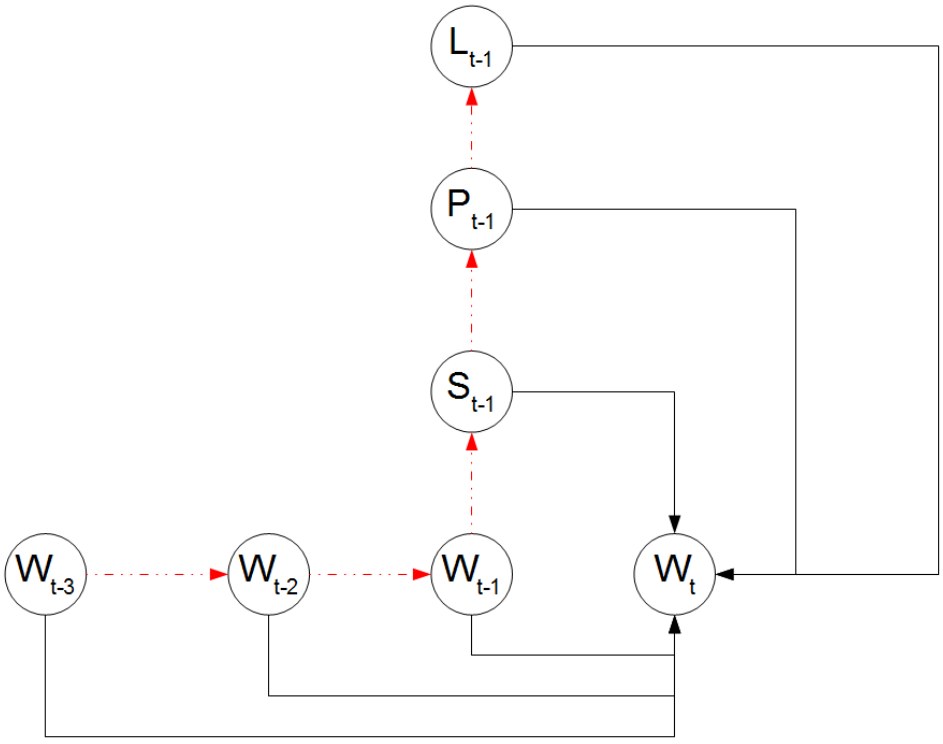

Figure 4.13: The design of our FLM labeled LIDPOSProb with words (W), LID (L), POS tags (P) and code-switch probabilities (S).

The graph to our FLM LIDPOSProb is shown in Figure 4.13. The model corresponding to this FLM is

$$p(W_t|W_{t-1}, W_{t-2}, W_{t-3}, L_{t_1}, P_{t_1}, S_{t-1}) \tag{4.7}$$

so the probability $p(w_t)$ is estimated with the language tag of the preceding word, one preceding POS tag, one preceding code-switch probability and three preceding words. This is indicated by the black connectors in Figure 4.13. The dotted red connectors show the backoff path, in case of insufficient data. We dropped the word history first, starting with the most distant one, afterwards dropping the code-switch probability, subsequently the POS tag and finally the language id leaving only the current word w_t. Dropping temporally distant words first is common for normal n-gram language models. We decided to drop the word W_{t-1} before any non-word factor because a word may be OOV, while the training data contains all possible values for LID, POS tags and code-switch point probabilities. The drop order of L_{t-1}, P_{t-1}, and S_{t-1} was based upon the MER performance improvements we achieved on our designed FLMs, which contained one of these parents in addition to common words (LID1, POSU, Prob2). Since we achieved the highest MER improvements applying the language ID, we dropped this factor at last. Since the code-switch probabilities performed worst, we drop this factor before POS tags.

The structure of the FLMs LID1, POSU, POSEN, Prob2, Prob3, Prob5 and LIDPOS is similar to the structure of LIDPOSProb, but one or two non-word nodes are missing. The structure of our FLM LID3 is a little different since there are three

preceding LIDs to consider not just the LID corresponding to the directly preceding word W_{t-1}. We are dropping the factors, which have the largest distance from the current word first and dropping the word before the LID. The FLM specification file for our FLM LID3 is the following:

```
1
W : 6 W(-1) W(-2) W(-3) L(-3) L(-2) L(-1) train.count train.lm 7
W1,W2,L1,L2,L3,W3 W3 ukndiscount gtmin 0 interpolate
W1,W2,L1,L2,L3 L3 ukndiscount gtmin 0 interpolate
W1,L1,L2,W2 W2 ukndiscount gtmin 0 interpolate
L1,L2,W1 L2  ukndiscount gtmin 0 interpolate
L1,W1 W1 ukndiscount gtmin 0 interpolate
L1 L1 ukndiscount gtmin 0 interpolate
0 0 ukndiscount gtmin 0 interpolate
1
```

4.5.3 N-Best Rescoring Parameters

In addition to using different FLMs, there are other parameters of importance for N-best rescoring:

- Number of hypothesizes extracted from the lattice per utterance: This is an important parameter since we are reranking utterances. More utterances to rerank may give the FLM the opportunity to increase the performance. However, more hypotheses increase the time needed for the rescoring process. We mainly worked with 100 hypotheses per utterances but also did some experiments with 10,000 hypothesizes per utterance. Please note that if the lattice is small, it may not be possible to extract the desired number of hypothesizes. This parameter is exclusively for N-best rescoring and does not affect the baseline.

- Language model weight (LZ): This defines the weight, which the language model score has within the overall score each individual hypothesis receives. A higher value increases the importance of the language model. We used 27, 30, 33, 37, 40, and 43 as language model weights. These values were chosen, considering the baseline performance on different LZ parameters on the development set.

- Word transition weight (LP): The word transition weight assesses the number of words in the hypothesis in relation to the acoustic model score. A higher word transition weight means a higher penalty to insert a word. We used -12 to 6 as LP values with an increment of 2.

4.5.4 FLM N-Best Rescoring Framework

The FLM N-best rescoring framework models the complete rescoring process with FLMs. We used mainly Shell Script and Perl for programming this framework. The framework requires the following preconditions:

- The baseline speech recognition system is created and decoded. The lattices are available in HTK [42] format.
- The FLM, which is to be applied for the rescoring process, is built on correctly formatted training data.
- The SRI Language Modeling Toolkit [15] is available.
- Since we programmed the framework using Perl and Shell Script, they both need to be available.
- To use an FLM with POS tag, the Stanford POS tagger needs to be available.
- If CS point probabilities are applied, they have to be estimated before.

The rescoring framework is displayed in Figure 4.14. The detailed process works in the following way:

1. As long as there are unprocessed language model weight and word penalty combinations, we perform an N-best rescoring with an unprocessed combinations. If there are no unprocessed combinations, we are done.
2. With the SRI Language Modeling Toolkit [15] we obtain an N-best list for each utterance in the lattice. For our experiments, we created lattices with (up to) 100 or 10,000 hypotheses per utterance.
3. We extract the cleaned hypotheses from the N-best list (which also includes acoustic model score, language model score, number of words and beginning and end tags) to convert the hypothesizes into the FLM training data format.
4. We convert the hypotheses to the FLM training data format so that the FLM can provide accurate language models scores. The training data format may include language, POS tag and code-switch point probability in addition to words. The language of a word is determined by evaluating the corresponding number of the characters of words. Since English words consist of ASCII characters and Mandarin words consists exclusively of non-ASCII characters, we can determine the language without error. For the POS tagging, we apply monolingual Mandarin and English taggers and map the tags to a unified set. Details can be found in Section 4.1. The code-switch point probability is estimated on the training set and written to a file. For the rescoring process, this file is read and the probability class is added into the hypotheses.
5. The hypotheses in FLM training data format are merged with the original N-best list to create a new N-best list containing acoustic model score, language model score of the baseline n-gram language model, number of words, and the hypotheses in FLM training data format.
6. Using the SRI Language Model Toolkit with the FLM extension, we apply the FLMs on the merged N-best list in order to obtain the language model scores of the FLM for each hypothesis.

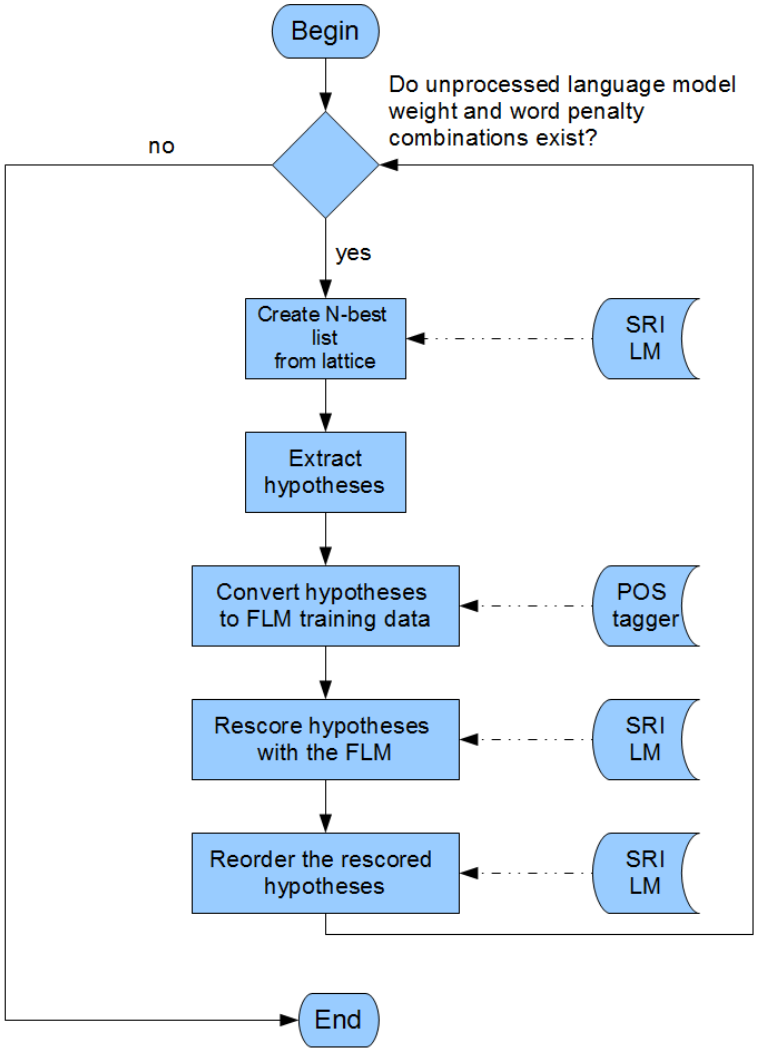

Figure 4.14: N-best rescoring framework applying factored language models.

7. With the SRI Language Model Toolkit, we rerank the different rescored hypotheses for each utterance, where the best ranked hypotheses for each utterance is used is compared with the reference.

8. Continue with step 1.

Figure 4.15 shows an example for the FLM rescoring framework of an N-best list with five utterances. In the example, the hypothesis ending with *RIGHT* gets a higher position due to the FLM rescoring. However, the order of the best hypothesis in this example remains unchanged. So we do not have a change in the performance here.

N Best List:
-7621.89 -6.576 4 我 要 吃 眼睛
-7554.93 -10.981 5 我 要 吃 眼睛 CONTACTS
-7550.13 -11.284 5 我 要 吃 眼睛 CARE
-7562.28 -10.806 5 我 要 吃 眼睛 THERE
-7574.24 -10.454 6 我 要 吃 眼睛 RIGHT

Cleaned hypotheses:
我 要 吃 眼睛
我 要 吃 眼睛 CONTACTS
我 要 吃 眼睛 CARE
我 要 吃 眼睛 THERE
我 要 吃 眼睛 RIGHT

FLM training data formatted hypotheses:
我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0
我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CONTACTS:P-NN:L-en:S-CSMIS
我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CARE:P-NN:L-en:S-CS1
我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 THERE:P-NN:L-en:S-CS0
我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 RIGHT:P-NN:L-en:S-CS0

Merged FLM training data formatted n best list
-7621.89 -6.576 4 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0
-7554.93 -10.981 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CONTACTS:P-NN:L-en:S-CSMIS
-7550.13 -11.284 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CARE:P-NN:L-en:S-CS1
-7562.28 -10.806 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 THERE:P-NN:L-en:S-CS0
-7574.24 -10.454 6 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 RIGHT:P-NN:L-en:S-CS0

FLM Rescored N Best List
-7621.89 -6.38232 4 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0
-7554.93 -10.495 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CONTACTS:P-NN:L-en:S-CSMIS
-7550.13 -11.2544 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CARE:P-NN:L-en:S-CS1
-7562.28 -11.3361 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 THERE:P-NN:L-en:S-CS0
-7574.24 -9.25854 6 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 RIGHT:P-NN:L-en:S-CS0

FLM Reordered N Best List
-7621.89 -6.38232 4 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0
-7574.24 -9.25854 6 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 RIGHT:P-NN:L-en:S-CS0
-7554.93 -10.495 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CONTACTS:P-NN:L-en:S-CSMIS
-7550.13 -11.2544 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 CARE:P-NN:L-en:S-CS1
-7562.28 -11.3361 5 我:P-PN:L-ch:S-CS0 要:P-VV:L-ch:S-CS0 吃:P-VV:L-ch:S-CS0 眼睛:P-NN:L-ch:S-CS0 THERE:P-NN:L-en:S-CS0

Figure 4.15: N-best rescoring example in different steps.

4.5.5 Evaluation of FLM Designs using N-Best Rescoring

We integrate our FLMs with N-best rescoring and calculate the MER and the perplexity on the development set and evaluation set. This subsection is about experiments with (up to) 100 hypotheses per utterances. We always consider the best LZ/LP combination for the MER.
Considering the MER results in Table 4.10 and 4.11, we can observe that applying a suitable FLM for N-best rescoring, leads to a minor performance improvement. Given the performance improvement, the language of the corresponding word seems to be the most useful factor. Regarding the MER, it does not seem to be relevant, whether we use three LID parents (LID3) or one LID parent (LID1). However, on later experiments with up to 10,000 hypotheses per utterance, our LID1 systems perform better than the corresponding LID3 system. The unified POS tags (POSU)

FLM	MER	Rel. Performance Improvement	LZ	LP
none (baseline)	59.1%	0%	37	-2
LID1	58.8%	0.5%	33	-10
LID3	58.8%	0.5%	37	-12
POSU	59.0%	0.2%	40	2
POSEN	59.2%	-0.2%	37	-4
Prob2	59.3%	-0.3%	40	-2
Prob3	59.4%	-0.5%	40	4
Prob5	59.5%	-0.7%	40	-6
LIDPOS	**58.6%**	**0.8%**	33	-12
LIDPOSProb	58.8%	0.5%	37	-10
GA	59.2%	-0.2%	37	-10

Table 4.10: MER result overview for N-best rescoring on the development set with 100 hypotheses per utterance

FLM	MER	Rel. Performance Improvement	LZ	LP
none (baseline)	60.8%	0%	43	-12
LID1	60.6%	0.3%	40	-12
LID3	60.6%	0.3%	37	-12
POSU	61.1%	-0.5%	43	-10
Prob2	61.0%	-0.3%	37	-10
LIDPOS	**60.5%**	**0.5%**	43	-10
LIDPOSProb	60.7%	0.2%	43	-10

Table 4.11: MER result overview for N-best rescoring on the evaluation set with 100 hypotheses per utterance

FLM	MER	Perplexity	Rel. Perplexity Reduction	Vocab
none (baseline)	59.1%	332.0	0.0%	2338
LID1	58.8%	285.1	14.1%	2340
LID3	58.8%	298.2	10.2%	2340
POSU	59.0%	381.2	-14.8%	2360
POSEN	59.2%	**279.4**	**15.8%**	2371
Prob2	59.3%	305.2	8.1%	2341
Prob3	59.4%	313.2	5.7%	2342
Prob5	59.5%	326.1	1.8%	2344
LIDPOS	**58.6%**	317.4	4.4%	2362
LIDPOSProb	58.8%	294.4	11.3%	2365

Table 4.12: MER and perplexity result overview for N-best rescoring on the development set with 100 hypotheses per utterance.

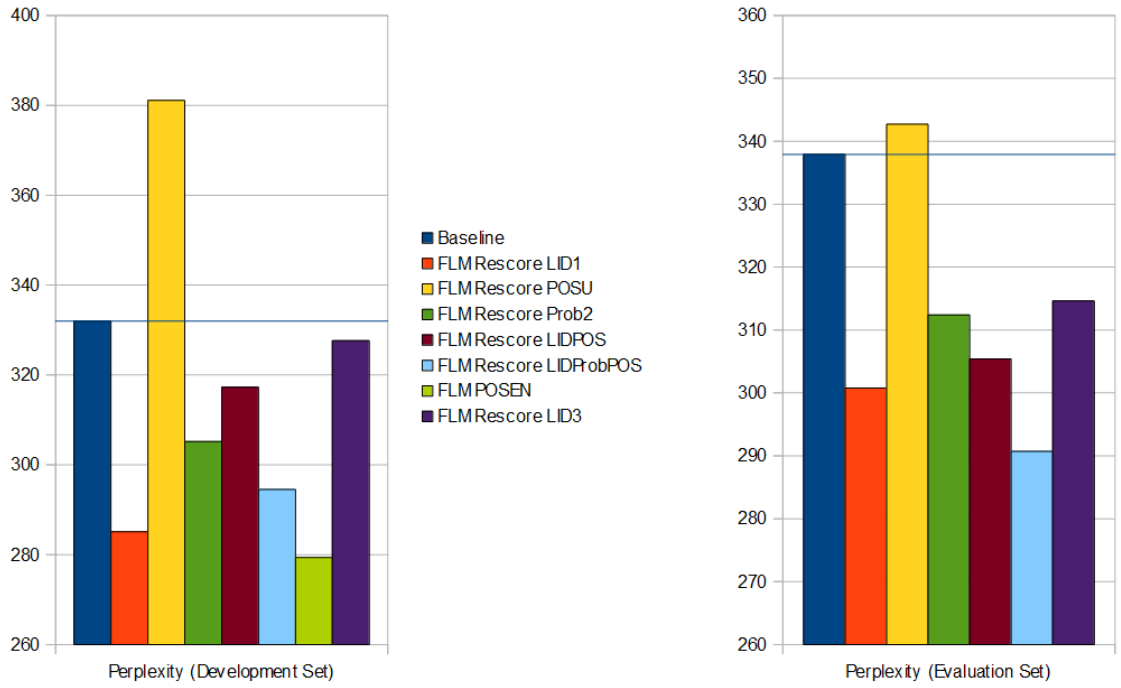

Figure 4.16: Perplexity result of FLM N-best rescoring experiments.

FLM	**MER**	**Perplexity**	**Rel. Perplexity Reduction**	**Vocab**
none (baseline)	60.8%	337.9	0.0%	2338
LID1	60.6%	**300.8**	**11.0%**	2340
LID3	60.6%	314.6	6.9%	2340
POSU	61.1%	342.4	-1.3%	2360
Prob2	61.0%	312.4	7.5%	2341
LIDPOS	**60.5%**	305.7	9.6%	2362
LIDPOSProb	60.7%	290.7	14.0%	2365

Table 4.13: MER and perplexity result overview for N-best rescoring on the evaluation set with 100 hypotheses per utterance.

seem to provide a minimal performance increase compared to the baseline on the development set, while the POSEN rescoring achieves a worse performance than the baseline. The code-switch probability decreases the MER performance, providing worse results with more different classes. We achieve the best performance with up to 100 hypotheses per utterance, if we utilize an FLM with words, LID and POS tags as factors (LIDPOS). Using this FLM, we achieve a performance improvement of 0.5% absolute, which is 0.8% relative one the development set. On the evaluation set, we can gain a performance improvement of 0.3% absolute, which is 0.5% relative. Rescoring with the FLM with all factors (LIDPOSProb) is not as good as applying the LIDPOS systems, since it seems that the code-switch probability decreases the performance. The FLM built with GA-FLM [41] (GA) provides a worse performance than our baseline. The best LZ/LP combinations vary for the different experiments, but the best LZ value seems to be 33, 37 or 40 while the best LP values are -10 and -12.

Beside the MER, we can evaluate the perplexity of various FLMs on development and evaluation set. Table 4.12, Table 4.13 and Figure 4.16 show the perplexity of our FLMs. Please recall that with OOV rate is at 7.76% on the development set and 8.14% on the evaluation set, considering words. All possible values for LID, POS tag and code-switch point probability are completely covered in our training data. Each FLM still applies the 2338 different words from the training transcriptions as vocabulary. In addition FLMs containing LID nodes use two language IDs in addition. FLMs applying unified POS tags, contain 22 POS tags in addition. The FLM POSEN contains 33 additional POS tags. FLMs incorporating code-switch point probabilities contain three to six code-switch point probability classes as additional entires. The exact number depends on the number of code-switch point probability classes applied. From the perplexity results on the development set, we can conclude that the language ID and the code switch-point probability reduce the perplexity of the FLM compared to the baseline. A lower number of code-switch point probabilities seems to lead to a greater perplexity reduction than using more classes. In combination with the MER results, it can be concluded that the code switch probability factor is more useful with less different classes. For the language ID, we can reach a greater perplexity reduction, if we only keep the language ID of the directly preceding word and not of three preceding words. This result indicates that the additional information, we can derive with further language ID parents, is outweighed by the more difficult estimation. For the unified POS tags (POSU), it can be observed that we get a higher perplexity than the baseline, while the POSEN approach provides the highest perplexity reduction for the development set. The POSEN system leaves the POS tags of the Mandarin tagger unmapped and tags all English words as “EN”. This is actually an language ID like information, about which tagger is used for each word, combined with further tagging information for Mandarin. This approach is problematic for our language islands based POS tagging as Chinese words may get the EN tag if the English tagger tagged them. Consequently, English words may receive a tag from the Mandarin tag if they have been tagged by a Mandarin tagger. These may reduce the difficulty of estimation and therefore the perplexity. But it makes the language model inaccurate to a small extend which explains the drop in MER performance. It is possible to change the POS tagging approach in a way that we tag every word with a tagger in the language the word belongs to. However, our POS taggers can handle a small amount of

foreign words. More information about the POS tagging approach can be found in Burgmer's diploma thesis [24]. We mainly used our POSU approach, were we used a unified mapping for the POS tags, where it is not necessary that every word is tagged with a tagger of the language the word belongs to. The POSU approach has a very high perplexity. In our experiments we realized that this high perplexity is related to the unified tag set and the handling of English words. However, this POS tagging approach leads to a small performance increase for the MER, also in combination with other factors. The POSU system on the evaluation set is an outlier for this observation.

Analyzing the impact of our CS probability factor, we derive that these factors have a negative impact on the MER, where the impact is worse with more different classes (Prob2, Prob3, Prob5). While increasing the number of different classes also increases the perplexity, the perplexity of the FLMs Prob2, Prob3 and Prob5 is still lower than the one of the baseline. This perplexity reduction can be explained by errors in the code-switch probability factor. For our development set we estimate our factors on the training set. These errors account for the case, that a particularly word may be assigned to a particularly code-switch point probability class on the training set. This class is assumed to be the accurate class on the development set as well. If this assumption is incorrect, we get an error. Given our estimation, the error on the development set is 7.0% for two classes, 10.4% for three classes and 13.8% for five classes. It is reasonable that the error rate increases with more classes. Furthermore the increasing error rate is at least partially responsible for the rising MER and rising perplexity from Prob2 to Prob3 and Prob5 and may explain the performance loss compared to the baseline as well. Evaluating the combination of factors, it is conclusive that the effects of the different factors, considering MER and perplexity are accounted for in FLMs, containing different non-word factors. The MER of our LIDPOS system is our best result.

4.5.6 Effect of the Number of Generated Hypotheses per Utterances

In our previous experiments we created (up to) 100 hypotheses per utterance for reordering. We applied the SRI language model toolkit [15] to derive the hypotheses from the lattices. Using 100 hypotheses per lattice has the advantage that the rescoring process is relatively fast. We use one core of 2.66 GHZ processor with 8GB random access memory. Then we obtain rescoring results in about two hours if our FLM does not need a POS tagger or in about 20 hours if we need a POS tagger to process. This time is for our complete language model weight and word penalty combinations which means 60 runs. However, if we create more hypotheses, we may get better results because more hypotheses mean that it is likely that we get can reduce the error rate, considering we chose the best hypothesize for each utterance. This statement is supported by experiments described in the next subsection. To investigate the effect of the number of hypotheses on performance and runtime, we increase the limit for the maximum of hypotheses from 100 to 10k. Investigating the runtime, it takes about 12 hours for a complete rescoring, if we do not need a POS tagger or about three weeks if we utilize a POS tagger. For obtaining this information we used one core of of 2.66 GHZ processor with 8GB random access memory and our complete LZ/LP list which means we do 60 runs., since we have six different

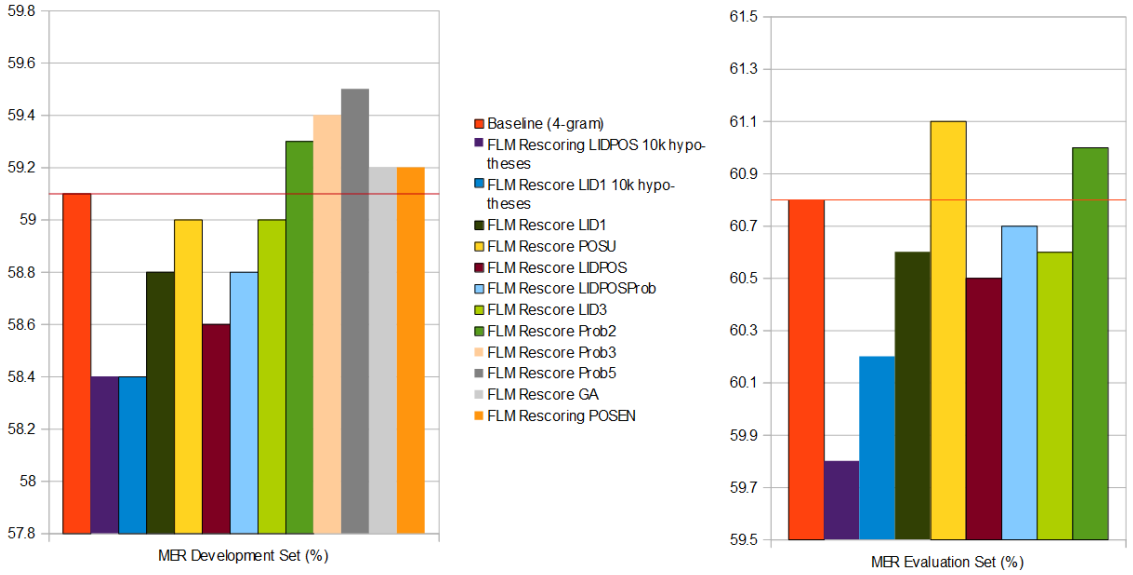

Figure 4.17: MER result of FLM N-best rescoring experiments with 100 and 10k hypotheses.

LZ and ten different LP values. We built our LID1, LID3, and our LIDPOS system with (up to) 10k hypotheses per utterance and obtained the MER results. As for our 100 hypotheses system, we always consider the LZ/LP combination with the lowest error rate. Our results are shown in Table 4.14. Figure 4.17 shows an overview of the MER and perplexity in our FLM N-best rescoring experiments with 100 and 10k hypotheses. Please note that our baseline does not apply any N-best rescoring and is therefore independent of the number of hypotheses. If we consider the results for our LIDPOS system, which provides the greatest performance increase, we see that we can reduce our MER further by applying up to 10k hypotheses per utterance compared to 100. For our development set, we can relatively reduce the MER by 1.18 % with 10k hypotheses, while we can only achieve a reduction of 0.85% applying 100 hypotheses. There is a stronger difference of the MER reduction on our evaluation set where we can achieve a 1.64% reduction with (up to) 10k hypotheses per utterance, while we can only achieve a 0.49% relative reduction using 100 hypotheses per utterance. We can more than triple our relative MER reduction on our evaluation set and reduce the error further for our development set. Therefore, applying (up to) 10k hypotheses per utterance definately has its use, given the higher runtime is unproblematic.

FLM	MER 100	#Hypos	Set	Rel. MER Gain	LZ	LP
none (baseline)	59.1%	-	Dev.	0.00%	37	-2
LID1	58.8%	100	Dev.	0.51%	33	-10
LID1	**58.4%**	10k	Dev.	**1.18%**	33	-10
LID3	58.8%	100	Dev.	0.51%	37	-12
LID3	58.6%	10k	Dev.	0.85%	37	-4
LIDPOS	58.6%	100	Dev.	0.85%	33	-12
LIDPOS	**58.4%**	10k	Dev.	**1.18%**	33	-10
none (baseline)	60.8%	-	Eval.	0.00%	40	6
LID1	60.6%	100	Eval.	0.33%	40	-12
LID1	60.2%	10k	Eval.	0.99%	43	-8
LID3	60.6%	100	Eval.	0.33%	37	-12
LID3	60.5%	10k	Eval.	0.49%	43	-6
LIDPOS	60.5%	100	Eval.	0.49%	43	-10
LIDPOS	**59.8%**	10k	Eval.	**1.64%**	43	-12

Table 4.14: MER result overview for (up to) 100 and 10k hypotheses (hypos) per utterance in the rescoring process.

System	Set	MER 100	MER 10k	Gain 100	Gain 10k
Baseline	Dev.	59.1%	59.1%	0.0%	0.0%
FLM LIDPOS	Dev.	58.6%	58.4%	0.5%	0.7%
Oracle	Dev.	50.8%	45.7%	8.3%	13.4%
Baseline	Eval.	60.8%	60.8%	0.0%	0.0%
FLM LIDPOS	Eval.	60.5%	59.8%	0.3%	1.0%
Oracle	Eval.	50.9%	46.2%	9.9%	14.6%

Table 4.15: Comparison of the baseline MER, the MER of FLM LIDPOS, and the oracle MER for 100 and 10k hypotheses.

4.5.7 Oracle Experiment

To further evaluate the results we performed an oracle experiment, i.e. we analyze, what the best MER would be, if we always pick the hypothesis with the lowest error rate for each utterance. This error rate is dependent on our baseline system, the number of hypotheses per utterance, and the chosen LZ/LP combination. These three factors influence the available hypothesis to calculate the MER compared to the reference. This experiment is not applying any FLM to rescore. We selected the LZ/LP parameter combination, which provides the best results. If we apply 100 hypotheses per utterance, the best combination is LZ=43 and LP=-4 and for 10k hypotheses per utterance LZ=43 and LP=2. These values were equal for the development set and evaluation set.

Table 4.15 shows the MER performance of our baseline, our best performing FLM (LIDPOS) and the oracle experiment for 100 and 10k hypotheses per utterance. Furthermore the absolute gain compared to the baseline is displayed. The LIDPOS FLM reaches a fraction of 3.03% to 6.85% of the gain, the corresponding oracle

experiment achieves. The fraction is 6.02% for 100 hypotheses and 5.22% for 10k hypotheses on the development set. Regarding the evaluation set the fraction is 3.03% for 100 hypotheses and 6.85% for 10k hypotheses. There are two reasons explaining our limited performance gain of our LIDPOS FLM compared to the oracle experiment:

- As described earlier, the ranking of a hypothesis depends on its acoustic model score, language model score, the number of words in the hypothesis, and the weighting of these factors with language model weight and word penalty. Since we do not make changes to the acoustic model score, we cannot expect to get a performance anywhere near the oracle.
- Our best performing FLM (LIDPOS) is not close to an ideal language model for our data.

For these reasons, it would be unrealistic to achieve the same performance as the oracle by applying FLMs. However, calculating the percentage of our improvement compared to the result of the oracle experiment may be more fair, compared to the relative reduction of the MER to 0, which we cannot achieve by N-best rescoring.

4.5.8 Significance of our Results

A key aspect of evaluating results is to investigate, if the improvements are significant. First, we will briefly describe very basic information about statistical significance. For this purpose, we translate several definitions from [43] into English and combine them with the application of our experiments:

- **Alternative hypothesis**: The alternative hypothesis μ means that there is a difference or connection between phenomena or approaches. In our case, our alternative hypothesis is that N-best rescoring with suitable FLMs on intra-sentential CS data leads to a performance improvement regarding the MER, compared to the baseline system.
- **Null hypothesis**: The null hypothesis μ_0 indicates that the connection proposed in the alternative hypothesis is not there. For our experiments the null hypothesis is the following: Applying N-best rescoring with suitable FLMs on intra-sentential CS data does in general not lead to a performance improvement regarding the MER compared to the baseline system.
- **Type I and type II errors**: A type I error means that a correct null hypothesis is denied and an incorrect alternative hypothesis is confirmed. A type II error indicates that a wrong null hypothesis is maintained.
- **Significance level**: The significance level α indicates an a-priori probability, which is determined by the researcher. This probability implies that the rejection of the null hypothesis, done by a significance test leads to a type I error. In particular, it can be said that the highest acceptable probability for an incorrect denial of a correct null hypothesis is α. Typical values for α are 5% , 1% , 0.1% and in rare cases 10%. The selected value depends on the gravity of a type I error. For example, if the engineers of a plane say that with

Baseline MER	Best FLM MER	#Hypos	Set	Error prob.
59.1%	58.6%	100	Dev.	6.8%
60.8%	60.5%	100	Eval.	29.3%
59.1%	58.4%	10k	Dev.	6.7%
60.8%	59.8%	10k	Eval.	5.1%

Table 4.16: Significance results for our FLM rescoring results.

a probability of 5%, the plane would crash, this would be very problematic. Even a 0.1% probability of such desaster may be to much. Given the fact that for our experiment, the gravity of an error is not that severe, we decide on a value of 5% for α, particularly because 5% is a standard error in case of less serious consequences.

To evaluate if our improvements are significant, we applied the t-test described in [43]. We used a paired t-test working on the MER of our baseline and our best corresponding FLM rescoring experiment for each utterance. We intend to show that our rescored FLM performs better than the baseline. Consequently, we can specifiy our hypothesis as $\mu > \mu_0$. We calculate the significance level, which we achieve with our experimental result. Our results are significant, if our error probability is smaller than $\alpha = 5\%$.

Table 4.16 shows our results. Our results would be significant if the error probability would be below 5% for a significance level of 5%. Since our error probability is higher than 5% for all experiments, the results are not significant on a lvl of 5%. However, with the exception of our result on the evaluation set applying 100 hypotheses per utterance, our results are statistically significant on a 10% level.

4.5.9 Summary of our FLM experiments

We investigated the application of FLMs for Mandarin-English speech data with intra-sentential CS. The idea behind using these models was to gain information, which are useful for CS. For FLM factors, we applied the language of a word, POS tags and a classified code-switch point probability in addition to words. We utilized FLMs in a N-best rescoring process. Considering our best language model which contains language ID , POS tags and words as factors, we can reduce our MER from 59.1% to 58.4% on the development set and from 60.8% to 59.8% on the evaluation set. A relative improvement of 1.18% and 1.64%, respectively is achieved. We conclude that utilizing additional information sources, like the language ID and POS tags, may be useful to reach performance improvements on our Mandarin-English CS data. Our results show that improvements can be obtained by adding language ID and POS tags as information sources to (factored) language models.

4.5.10 Limits of this Work and Future Prospects

In this subsections, we describe the limits of this work and direct into future prospects.

Limits of our work and future work are as follows:

- **Use additional factors:** Additional factors can be added to an FLM for data with intra-sentential CS. Since a common reason for CS is that words are more available in one language, features related to the language dependent word availability, difficulty and frequency of use could be added. Using features related to the word semantic, like described in [19], would also be an option.

- **Language model interpolation:** One of the limits of this work is that all created FLMs are exclusively built on the small multilingual training text data, where we remain with an OOV rate of 7.76% and 8.14% on development and evaluation set. We did not work with interpolated FLMs since the current version of the SRI language modeling toolkit [15] does not support writing FLMs. Frequent interpolation of FLMs from larger text corpora has very negative effects on the runtime. For future work, it would be reasonable to include support for writing FLMs into the SRI language modeling toolkit and develop suitable means for FLMs and language model interpolation. Adressing this issue would be particularly helpful, if suitable monolingual conversational and meeting data becomes available. Please note that the English text data, we had at our disposal was quite different from conversational data.

- **CS speech database:** A key limit of this work is that we only had 163 minutes of speech data with Mandarin-English CS available, with only one speaker with a reasonable speaking time. While this data was enough to perform speaker adaptation, it is not enough to build a multilingual acoustic model with English-Mandarin training data. While this may not be necessary, it is to consider that all our results are speaker-dependent. However, the speaker-dependent CS corpus recorded at the Hong Kong University of Science and Technology was the only CS corpus, which was available from the beginning of our work. The Mandarin-English SEAME [44] corpus, which was recorded in Singapore and Malaysia should be a solution to this problem.

- **CS text data:** Another issue is the limited availability of CS Mandarin-English text data. This is related to the nature of CS, since it is often used spontaneously and rarely used in (formal) writing. CS text data may be generated cross-lingually and artificially with the help of machine translation tools. We hope to get a significant performance boost with large code-switched language model training data available. This work may be combined with an ongoing diploma thesis at the Cognitive Systems Lab by Blaicher [45], dealing with artificial code-switch corpus generation. Another possibility regarding this issue is to extend the functionality of tools, which crawl the internet for language model data such as our Rapid Language Adaptation Toolkit (RLAT) [46] [47], to search for multilingual code switching data. For example, in online chats, it may be possible to obtain suitable data with CS. Furthermore the RLAT language identification engine could be extended to detect websites containing CS. However, currently we do not know, if there is a relevant amount of websites containing CS on the world wide web.

- **Direct application of FLMs in the decoding process:** An extension of a decoder in a way that it can directly apply CS should result in a more significant performance increase, than our N-best rescoring experiments, since

N-best rescoring is strongly depending on the baseline language model. Extending a decoder is a challenging and time consuming task, which was not the primary focus of this diploma thesis.

- **Extend history of code-switch probabilities using additional text data:** Due to the lack of sufficent CS training text data, we created our code-switch probabilities based upon trigger words (unigrams). Extending the CS point probability from trigger words to trigger phrases (n-grams) may be more effective. This extension should be done, if more CS text data is available.

- **CS POS tagger:** The POS tagging with monolingual taggers for bilingual data with intra-sentential CS is not optimal. We applied Burgmer's approach [24] to utilize available, monolingual taggers and distributed phrases to them. Creating a truly bilingual tagger should lead to more suitable POS tags and hopefully to further performance improvement.

- **Improving baseline performance with acoustic model and dictionary:** While conversational and meeting data with CS is difficult to recognize, it may be possible to get a significant performance improvement on the acoustic model and the dictionary. Combining acoustic model, dictionary and language model improvements for speech data containing code-switches should lead to stronger improvements than exclusively working on one of these components. Nevertheless, this work mainly faced language model issues.

- **Extending the work to new languages:** CS does not only happen between Mandarin and English. Therefore, it is interesting to investigate different language pairs for CS and automatic speech recognition. Other examples of CS, are English and Spanish [1] and Hindi and English [21].

- **Including a language boundary detection system in the ASR process:** The results of other work for language identification and language boundary detection indicates that providing additional information about language boundaries to the speech recognizer is a way to improve the performance of speech recognition on CS.

5. Summary

In this thesis we have investigated the use of additional text based features for automatic speech recognition (ASR) on Mandarin-English speech data containing intra-sentential code-switching (CS).

We worked on English-Mandarin conversational meeting speech data with 163 minutes length.

In the first part, we explored the use of different features for multilingual language modeling: We predict language and CSP. Our results were evaluated on a development set and an evaluation set. The development set was used to select suitable classifiers for our prediction experiments, and to evaluate factored language model designs, and speech recognition parameters. The text based prediction of the language of a word in an utterance was investigated. Language, part-of-speech (POS) tags and occurrence features of preceding words in the current utterance are applied. We obtained an F-measure for the prediction of the language of 0.926 on the development set and of 0.901 on the evaluation set, respectively. Based upon our experiments to predict language, we predicted CSPs. We obtained an F-measure of 0.205 on the development set and 0.300 on the evaluation set. Still we achieve an improvement compared to random predictions of CSPs, where we calculated an F-measure of 0.116 on our development set and 0.142 on our evaluation set. From our prediction experiments, we can conclude that it is useful to utilize language and POS tag information in the language model on CS data.

In the second part of this work, we included and evaluated POS tags, language identification (LID) and a class based CSP probability in the language model component for ASR. We derived POS features from our text data utilizing POS taggers, while the LID is processed rule based without error since Mandarin and English have different scripts. Words were split into classes based upon the likelihood, if a code-switch follows after each word estimated on our training text data. These classification was used as our CSP probability. Factored language models were applied to include these factors and followed by N-best rescoring. We achieved the highest improvement with factored language models containing LID and POS tags in addition to words as factors. For evaluating the results, we utilized a mixed error rate (MER), which is the character error rate (CER) for Mandarin and the

word error rate (WER) for English. With the application of our best performing factored language model, we reduced the MER from 59.1% to 58.4% on the development set and from 60.8% to 59.8% on the evaluation set, respectively. This is a relative improvement of 1.18% and 1.64%, respectively. Our results indicate that improvements can be achieved by adding language ID and POS tags as information sources to factored language models, while our code-switch point probability does not provide MER improvements.

Bibliography

[1] Grosjean F.: *Life with Two Languages. An Introduction to Bilingualism.* Harvard University Press. 1982.

[2] Schultz T., Kirchhoff K.: *Multilingual Speech Processing.* Academic Press. 2006.

[3] Bilmes J., Kirchhoff K.: Factored language models and generalized parallel backoff. *In Proceedings of HLT/NAACL, pages 4-6.* 2003.

[4] Gumperz J.: *Discourse Strategies.* Cambridge University Press. 1982.

[5] Joshi A.: Processing of sentences with intrasentential code-switching. *COLING-82f.* 1982.

[6] Sankoff D., Poplack S.: A Formal Grammar for Code-Switching. *Papers in Linguistics, International Journal of Human Communication, 14(1):3(46).* 1981.

[7] Kroeger P.: *Analyzing Grammar: An Introduction.* Cambridge University Press. 2005.

[8] Bokamba E.: Are there syntactic constraints on code-mixing? *World Englishes 8 (3): 277-92.* 1989.

[9] Bhatt R.: Code-switching and the functional head constraint. *Proceedings of the Eleventh Eastern States Conference on Linguistics.* 1995.

[10] Gebhardt J.: Multilingual Acoustic Model Combination using the Rapid Language Adaptation Toolkit (RLAT). student research project. Karlsruhe Institute of Technology. 2009.

[11] Young S.: Large Vocabulary Continuous Speech Recognition: A Review. 1996.

[12] Huang X., Acero A., Hon H.: *Spoken Language Processing, a Guide to Theory, Algorithm and System Development.* Prentice Hall. 2001.

[13] Rosenfeld R.: Two Decades Of Statistical Language Modeling: Where Do We Go From Here? *In Proc. IEEE.* 2000.

[14] Bellegarda J.: Statistical language model adaptation: review and perspectives. *Speech Communication 42 (2004) 93-108.* 2004.

[15] Stolcke A.: SRILM An extensible language modeling toolkit. *In Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002).* 2002.

[16] Schultz T.: Multilinguale Spracherkennung - Kombination akustischer Modelle zur Portierung auf neue Sprachen. doctoral thesis. University of Karlsruhe. 2000.

[17] Lyu D., Lyu R.: Language Identification on Code-Switching Utterances Using Multiple Cues. *In Proceedings of Interspeech.* 2008.

[18] Yeh C-F., Huang C-Y., Sun L-C., Lee L-S.: An Integrated Framework for Transcribing Mandarin-English Code-mixed Lectures with Improved Acoustic and Language Modeling. *Proc. International Symposium on Chinese Spoken Language Processing (ISCSLP) 2010, pp. 246-250.* 2010.

[19] Cao H., P.Ching , Lee T., Yeung Y.: Semantics-based Language Modeling for Cantonese-English Code-mixing Speech Recognition. *Proc. International Symposium on Chinese Spoken Language Processing (ISCSLP) 2010, pp. 246-250.* 2010.

[20] Tsai T-L., Chiang C-Y., Yu H-M., Lo L-S., Wang Y-R., Chen S-H.: A Study on Hakka and Mixed Hakka-Mandarin Speech Recognition. *Proc. International Symposium on Chinese Spoken Language Processing (ISCSLP) 2010, pp. 246-250.* 2010.

[21] Bhuvanagiri K., Kopparapu S.: An Approach to Mixed Language Automatic Speech Recognition. *Proc. Oriental chapter of COCOSDA (The International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques).* 2010.

[22] Witten I., Frank E.: *Data Mining - Practical Machine Learning Tools and Techniques.* Elsevier. 2005.

[23] Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I.: The WEKA Data Mining Software: An Update. *SIGKDD Explorations.* 2009.

[24] Burgmer C.: Detecting Code-Switch Events Based on Textual Features. diploma thesis. Karlsruhe Institute of Technology and Hong Kong University of Science and Technology. 2010.

[25] Finke M., Geutner P., Hild H., Kemp T., Ries K., Westphal M.: The Karlsruher-Verbmobil Speech Recognition Engine. *In Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, Munich.* 1997.

[26] Toutanova K., Manning C.: Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. *Proc. Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC),pp. 63-70.* 2000.

[27] Toutanova K., Klein D., Manning C., Singer Y.: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. *Proc. HLT-NAACL, pp. 252-259.* 2003.

[28] Toutanova K., Manning D., Morgan W., Tseng H., Rafferty A.: Stanfort POS Tagger v 1.6 Retrieved 28 September 2008, from http://nlp.stanford.edu/software/tagger.shtml.

[29] Marcus M., Marcinkiewicz M., Santorini B.: Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics,19(2).* 1993.

[30] Xue N., Xia F., Chiou F., Palmer M.: The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering, 11(2).* 2005.

[31] Solorio T., Liu Y.: Learning to Predict Code-Switching Points. *Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 973-981, Honolulu.* 2008.

[32] Solorio T., Liu Y.: Part of Speech Tagging for English-Spanish Code-Switched Text. *Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1051-1060, Honolulu.* 2008.

[33] Quinlan R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers. 1993.

[34] Cessie S., Houwelingen J.: Ridge Estimators in Logistic Regression. *Applied Statistics. 41(1):191-201.* 1992.

[35] Friedman J., R. Tibshirani T. Hastie: Additive Logistic Regression: a Statistical View of Boosting. *Stanford University.* 1998.

[36] John G., Langley P.: Estimating Continuous Distributions in Bayesian Classifiers. *Eleventh Conference on Uncertainty in Artificial Intelligence.* 1995.

[37] Chang C-C., Lin C-J.: LIBSVM - A Library for Support Vector Machines. *http://www.csie.ntu.edu.tw/ cjlin/libsvm/.* 2001.

[38] Hsiao R., Fuhs M., Tam Y-C., Jin Q., Schultz T.: The CMU-InterACT 2008 Mandarin Transcription System. *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing.* 2008.

[39] Kneser R., Ney H.: Improved backing-off for N-gram language modeling. *In Proc. ICASSP, 181-184.* 1995.

[40] Kirchhoff K., Bilmes J., Duh K.: Factored Language Model Tutorial. *UWEE Technical Report Number UWEETR-2008-0004.* 2008.

[41] Duh K., Kirchhoff K.: Automatic learning of language model structure. *In Proceedings of the International Conference on Computational Linguistics (COLING).* 2004.

[42] Young S., Evermann G., Kershaw D., Moore G., Odell J., Ollason D., Valtchev V., Woodland P.: *The HTK Book (for HKT version 3.4.1).* Cambridge University Engineering. 2009.

[43] Bortz J., Schuster C.: *Statistik fuer Human- und Sozialwissenschaftler.* Springer. 2010.

[44] Lyu D-C., Tan T-P., Chng E-S., H.Li : SEAME: a Mandarin-English Code-switching Speech Corpus in South-East Asia. *In Proc. Interspeech.* 2010.

[45] Blaicher F.: SMT-based generation for Code-Switching Language Models. diploma thesis. Karlsruhe Institute of Technology. 2011.

[46] Schultz T., Black A., Badaskar S., Hornyak M., Kominek J.: SPICE: Web based Tools for Rapid Language Adaptation in Speech Processing Systems. *Interspeech.* 2007.

[47] Rapid Language Adapation Toolkit (RLAT). http://csl.ira.uka.de/rlat-dev.