Karlsruhe Institute of Technology

Language Model Adaptation using Latent Dirichlet Allocation

Studienarbeit
of

Karl Karnadi

at the Institut for Anthropomatics
of the Fakulty of Informatics

July 18, 2012

Advisors
Kevin Kilgour

Contents

1 Introduction **1**
- 1.1 Objectives . . . 2
- 1.2 Outlines . . . 2

2 Overview **3**
- 2.1 Language model . . . 3
- 2.2 LM evaluation . . . 3
  - 2.2.1 Perplexity . . . 4
  - 2.2.2 Word Error Rate . . . 4
- 2.3 N-gram . . . 4
  - 2.3.1 Smoothing . . . 5
- 2.4 Related works . . . 6
  - 2.4.1 Trigger-based LM . . . 6
  - 2.4.2 Cache-based LM . . . 7
  - 2.4.3 LM Adaptation using Classification Techniques . . . 7
    - 2.4.3.1 LSA . . . 8
    - 2.4.3.2 pLSA . . . 8
    - 2.4.3.3 LDA . . . 8

3 Analysis & Design **13**
- 3.1 Data preparation . . . 13
- 3.2 The training . . . 15
  - 3.2.1 Variational Bayes-Expectation Maximization algorithm . . . 16
- 3.3 The testing . . . 16
  - 3.3.1 LM Adaptation . . . 16
  - 3.3.2 Interpolation with the n-gram model . . . 17

4 Implementation **19**
- 4.1 Janus Recognition Toolkit (JRTk) . . . 19
- 4.2 LDA library . . . 19
- 4.3 SRILM toolkit . . . 19
- 4.4 ARPA back-off file format . . . 20
- 4.5 FSM toolkit . . . 21
- 4.6 Scoring toolkit . . . 21

5 Evaluation **23**
- 5.1 Baseline System . . . 23
- 5.2 LDA-adapted System . . . 24
- 5.3 Experiments . . . 24

5.4 Results 24
5.4.1 Number of Iteration 25
5.4.2 Number of Topics 25
5.4.3 LM weights and word penalty 26
5.4.4 Further analysis 26

6 Summary **29**
6.1 Further works 29

Bibliography **31**

1. Introduction

Speech recognition is a task that comes naturally to every human since their childhood. We can talk about football, politics, economy, and we can switch fluently from one topic to the other. When we talk to someone, even in a noisy place like a train station or a party, we can still guess what people do speak about. Even before it was spoken, we have some expectation that certain keywords would be spoken more than other words. When we participated in a conversation about the US politics, for example, we expect that "Obama" would be more likely spoken rather than "Oh, bummer!", even when we couldn't differentiate the two meaning based on just hearing the audio alone. This feature helps us tremendously to correctly understand speech in our daily lives. When we build an automatic speech recognition (ASR) system this would brings us a new challenge.

In an ASR system, how closely the training data resembles the evaluation data influences the accuracy of the trained system. If we use training data from mostly football-related news, the ASR would be good in recognizing sentences spoken in similar shows, but would not perform well if we use it to recognize news about the US election. This grouping is often called "domain". One straightforward solution for optimization would be to build many small systems, one for each domain, and use them based on the domain of the evaluation data we encounter. There are serious challenges however, with this approach. Recognizing the domain of the evaluation data is a difficult task. The domain of a text data more resembles probability distribution rather than all-or-nothing. Analogous with the real life, we cannot say for certain that someone is talking about football and not politics, or vice versa. It could also be both, a mix between different domains where certain domains are stronger than the other. Furthermore, how many domains are there? Just football and politics? or more? Should we name them explicitly or can the grouping be done automatically? The next two chapters will describe an algorithm which promises to provide a solution to these challenges.

However, before we delve into details, it is helpful to define and describe first the basic knowledge of what an ASR is, how it works, and the different parts make up an ASR. An ASR is, simply defined, a system which converts spoken words into

words. An ASR system use the following equation to produce the best sequence of words as a recognition output:

$$\hat{W} = \arg\max_{W} P(W|X) = \arg\max_{W} \frac{P(X|W) \cdot P(W)}{P(X)} = \arg\max_{W} P(X|W) \cdot P(W)$$

We can see here that in order to get the best hypothesized words $\hat{W}$, we have to compose a sequence of words which matched the acoustic input X. In statistical term, it maximizes the conditional probability of the words W given the acoustic X. Using the Bayesian equation and further simplification, we get the above equation, where $P(X|W)$ represent Acoustic Modelling (AM) and $P(W)$ is represent Language Modelling (LM). In this work, we focus on an extension of a LM.

1.1 Objectives

The purpose of this work is to improve the recognition result of an English speech recognition system by using a Language Model Adaptation technique, in particular a model called Latent Dirichlet Allocation (LDA). The LDA model is built using the English data and afterwards the resulting model is applied to the current system which uses conventional n-gram model. By applying this model and test various parameter it is hoped that this can improve the recognition result.

1.2 Outlines

In chapter 2 the basics of the Language Modelling will me explained. Various techniques and related works will also be described and both of it's advantages and disadvantages will be analyzed. Chapter 3 contains the parameter and the data used for this experiment. Chapter 4 describes the design of this experiment. It's implementation will be described in Chapter 5, while the result will be evaluated in Chapter 6. The last chapter, chapter 7 will summarize the whole work and list some suggestions for future works.

2. Overview

This chapter is the overview of language modeling strategies that are well known today. At first the popular N-gram LM will be explained along with the challenges it still has and various techniques developed to solve them. In the last section the LDA and other language model adaptation techniques will be given.

2.1 Language model

As mentioned in Chapter 1, the language model (LM) is an important part of the speech recognition. Language model can be represented by a probability distribution $P(W)$ which indicates how often certain sequence of words or a sentence occurs in our data. For example in the English language, P(how are you) would be much larger than P(Twas bryllyg and ye slythy toves). This helps the recognition system immensely to decide which sequences of words should it choose to hypothesize, given the often very similar acoustic sound. It is also consistent with the fact that we also using something similar to language modelling in our daily life, guessing the content of a speech based on it's context or how make sense the sentence would be. $P(W)$ can be calculated using the following equation:

$$P(W) = P(W_1, W_2, \cdots, W_n) = P(W_1)\, P(W_2|W_1) \cdots P(W_n|W_1, W_2, \cdots, W_{n-1})$$

However, it is not practical to do so, since the lengthy conditional probability ($P(W_n|W_1, W_2, \cdots, W_{n-1})$ with a large n) would require large amount of data to estimate the value accurately. To solve this problem we use the n-gram technique, which simplifies the equation by considering not all words which appears before the current word, but only the last $(n-1)$ words.

2.2 LM evaluation

Before further discussing various LM methods, it is important to know how the quality of an LM is measured, so that we can compare different approaches or further improving the existing ones. There are two common approach on the measurement, perplexity (ppl) and word error rate (WER).

2.2.1 Perplexity

Given a set of test sentences $T = \{W_1, W_2, \cdots, W_N\}$, the probability of the test set can be calculated as:

$$P(T) = \prod_{i=1}^{N} P(W_i)$$

The cross-entropy of the model is defined as:

$$H_p(T) = -\frac{1}{W_T} log_2 P(T)$$

where W_T indicates the number of words which belongs to the test set T

The perplexity is calculated by the following equation:

$$\mathsf{ppl} = 2^{H_p(T)}$$

Perplexity indicates the predictive power of an LM toward a body of text. Lower perplexity values indicates a more predictive LM, and generally gives a better recognition result. This measurement approach has it's advantage for measuring the LM only, thus spare a lot of time and computing power. However, this correlation to the recognition result cannot be guaranteed, since a predictive LM doesn't necessarily predict a correct hypothesized words for decoding. Also, in the case of lattice rescoring, it is only possible to measure the quality in WER, not in perplexity.

2.2.2 Word Error Rate

When we want to test the quality of the whole recognition system, the word error rate (WER) is the commonly used techniques. The WER is calculated based on the minimal edit distance between the hypothesis and the reference sentence, which is the minimal numbers of subsitutions s, insertions i, and deletions d required to transform the hypothesis into the reference. The WER is calculated as follows:

$$WER = \frac{s + i + d}{n} \times 100\%$$

where n is the number of words in the reference sentence.

If used to evaluate an acoustic model, the WER has it's limitation. The similarly sounding words and the completely different words would both be equally evaluated as word errors. But for the purpose of evaluating LM-adaptation, the WER approach is more than enough.

2.3 N-gram

N-gram is the most applied statistical-based approach to build a language model. An n-gram in this context is a sequence of n words from a sentence. An n-gram model predicts conditional probability of a word W_i based only on the last $n-1$ words $W_{i-1}, W_{i-2}, ..., W_{i-(n-1)}$ or in probability terms:

$$P(W_i|h_i) \approx P(W_i|W_{i-n+1}...W_{i-1})$$

The n-gram has alternative names for n equals 2 (bigram), and 3 (trigram). Large amount of text is used to estimate this probability. We call the collection of text used to build an n-gram model as a training corpus. The accuracy of the model depends on both the quality of the corpus (the coverage) and the size, the larger the better. More complex n-gram requires more data. The n-gram can be estimated by counting the frequencies of the of the word pair $C(W_{i-n+1}...W_i)$ and $C(W_{i-n+1}...W_{i-1})$ and compare them as follows:

$$\hat{P}(W_i|W_{i-n+1}...W_{i-1}) = \frac{C(W_{i-n+1}...W_i)}{C(W_{i-n+1}...W_{i-1})}$$

So for example if we use a bigram model, the probability of the sentence J̈ohn read a bookẅould be

$$P(John\ read\ a\ book) = P(John|\texttt{<s>})\,P(read|John)\,P(a|read)\,P(book|a)\,P(\texttt{</s>}|book)$$

The <s> is normally used to denote sentence begin and </s> for sentence ending. We count each bigram by using the n-gram equation, for example:

$$P(read|John) = \frac{C(John\ read)}{C(John)}$$

where each of the Cs is calculated by counting how often a word or words (like J̈ohn read)̈ appear in the training corpus.

The advantage of the n-gram model is that it can capture daily used sentences (spontaneous speech), which are not necessarily correct grammatically. Furthermore the short distance grammatical features are automatically captured by the n-gram approach. However, there are several criticism toward n-gram which are addressed in the following sections.

2.3.1 Smoothing

Even when large corpus is used, many word sequences would appear very seldom or never appear at all. This leads to very small probabilities (close to zero) or zero probability. This problem is often called d̈ata sparsity.̈ To make estimated probabilities robust for unseen or rarely seen data, we use the techniques called smoothing. Most of the smoothing techniques can be classified into two: back-off or interpolated models.

The idea of back-off smoothing, is to solve the probability of unobserved n-grams by considering the lower order probability distribution. Back-off models uses this following equation:

$$P_{smooth}\left(W_i \mid W_{i-n+1}^{i-1}\right) = \begin{cases} \alpha\left(W_i \mid W_{i-n+1}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) > 0 \\ \gamma\left(W_i \mid W_{i-n+1}^{i-1}\right) P_{smooth}\left(W_i \mid W_{i-n+2}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) = 0 \end{cases}$$

$$\text{where } w_p^q = w_p...w_q$$

Here we can see that $P_{smooth}\left(W_i \mid W_{i-n+2}^{i-1}\right)$ is the back-off lower order distribution used, and $\gamma\left(W_i \mid W_{i-n+1}^{i-1}\right)$ is the scaling factor used. The lower order probability

only calculated whenever it is needed. Katz ([Katz87]) and Kneser-Ney ([KnNe95]) smoothing methods are commonly used for back-off models.

In contrary to back-off smoothing, interpolated smoothing always use lower order distribution for both observed and unobserved n-grams.

$$P_{smooth}\left(W_i \mid W_{i-n+1}^{i-1}\right) = \begin{cases} \alpha\left(W_i \mid W_{i-n+1}^{i-1}\right) & \\ +\gamma\left(W_i \mid W_{i-n+1}^{i-1}\right) P_{smooth}\left(W_i \mid W_{i-n+2}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) > 0 \\ \gamma\left(W_i \mid W_{i-n+1}^{i-1}\right) P_{smooth}\left(W_i \mid W_{i-n+2}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) = 0 \end{cases}$$

$$\text{where } w_p^q = w_p ... w_q$$

Jelinek-Mercer ([JeBM75]) and Witten-Bell ([WiBe89]) are commonly used interpolated models.

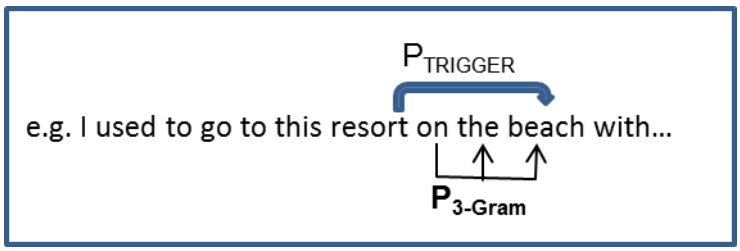
2.4 Related works

In this section further improvements of language model outside of n-gram will be discussed. Some of these techniques can be seen as a precendent for the LDA-based technique tested in this thesis.

Because of the data limitation and the relatively small n, the n-gram language model can only capture local context. In other hand, when we hear speeches in our daily lives, the words we hear are often dependent on the topics or context of the whole conversation. These can only be captured by looking at other words not only in the same sentence or the last (n-1) words, but also in the neighboring sentences. The following are approaches which try to capture those long-distance words and using them to build a new model. These model, which will be described in the following, are used to improve recognition result by combining it's usage with n-gram model.

2.4.1 Trigger-based LM

To extract information from the document history, the trigger-based LM uses the so-called *trigger pair* as the basic information bearing element. The $(A \rightarrow B)$ is called a *trigger-pair*, if a word sequence A is significantly correlated with the word sequence B, with A being the trigger and B the triggered sequence. So when A occurs, it will trigger B, changing it's probability estimation. The following is an example of how trigger-based LM works:



On this example, we can see that the word "resort" influences (triggers) the probability of the word "beach". At the same time the word "on" and "the" still influences "beach" as a 3-gram history.

Because of the large amount of trigger-pair possible (much larger than bigram vocabulary), firstly a filtering procedure must be done to select the most promising trigger pairs. Then the Maximum Entropy (ME) principle is used to combine different probability estimates into one estimate. The last step is mixing this model with the conventional n-gram model. Further details on implementing trigger-based model can be seen in [LaRR93].

This approach produced 12% lower perplexity compared to the conventional trigram, as tested in [LaRR93]. The advantage of this method is it's simplicity, but the disadvantages are the computational requirements.

2.4.2 Cache-based LM

Another approach is called cache-based LM [ClRo97]. It saves the recently occurred word in an exponentially decaying n-gram cache. The motivation behind cache-based LM is that words which have recently occurred in a piece of text have a higher probability of re-occurring. One would also expect that more recent words should contribute more the the probabilities of the cache. Thus, the cache is modeled in which the importance of each position in the cache decays exponentially along with the increase of the distance from the current examined word.

$$P_{cache}\left(w_i|w_1, w_2, ..., w_{i-1}\right) = \beta \sum_{j=1}^{i-1} I_{w_i=w_j} e^{-\alpha(i-j)}$$

where I is the indicator function such that $I_A = 1$ if A is true, and 0 otherwise, α is the decay rate and β is a normalising constant.

This cache probability is then weighted and combined with the probability from the n-gram approach, using the following equation:

$$P\left(w_i|w_1, w_2, \cdots, w_{i-1}\right) = \mu P_{cache}\left(w_i|w_1, w_2, \cdots, w_{i-1}\right) + (1-\mu) P_{trigram}\left(w_i|w_1, w_2, \cdots, w_{i-1}\right)$$

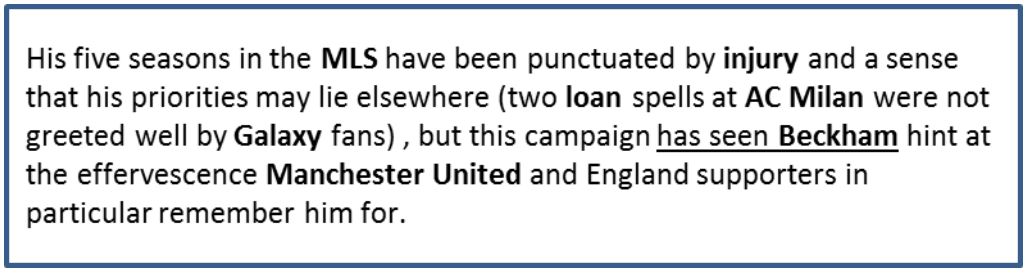
When uses 50 mixtures, this approach can achieve 24% decrease in perplexity. The advantage of this approach, is that it still performs better when the hypothesized transcription used for the adaptation has been somewhat corrupted. However, this technique is ideal for 2-pass approach used in lattice rescoring, and the rescoring must be done in a short-scale basis (for example: sentence-by-sentence), in order to keep the lattice's size small.

While the cache-based LM is using word surface form to model the word probabilities, the next approach uses a new concept called latent topics.

2.4.3 LM Adaptation using Classification Techniques

Instead of using general-purpose language models, the ASR system performs better when the language model used is smaller and in the similar domain with the the recognized speech. There are few techniques we can use which utilizes models developed for the text classification task. In these techniques, the domain of a text is modeled as a latent variable called topic and it can be automatically produced from documents.

Before dwelve further into the models and methods, consider the following example:



In this example the underlined words "has seen Beckham" indicates the 3-gram model being used, where the probability of the word "Beckham" depends only on the two previous words "has" and "seen": $P_{3-gram}\,(has\ seen)$. However, skimming on the paragraph would gives us information that the topic being discussed is about soccer. This is because there are so many soccer-related keyword on the paragraph. So for the next word, the soccer-related words like Beckham should have higher probabilities. This text feature (the topic of the text) uncaptured by the n-gram, can be captured by using the following text-modelling techniques.

2.4.3.1 LSA

Latent semantic analysis (LSA), also sometimes called Latent Semantic Indexing (LSI), is a technique in natural language processing in which the relationship between a set of documents and the terms they contain are modelled by a set of concepts related to the documents and terms. Let X be a matrix where element (i,j) describes the occurrence of term i in document j. LSA reduce the X matrix by using singular value decomposition (SVD) of the matrix to identify a linear subspace that capture most of the variance in the collection. It was tested on two data set, MED and CISI. MED was a commonly studied collection of medical abstracts, and the LSI shows 13% improvement (better precision) over raw term matching. The second data set CISI is a set of 1460 information science abstracts, which has been consistently difficult for automatic retrieval methods. In this second set, LSI does not show any improvement. This approach is based on the work by Deerwester in [DDFL$^+$90].

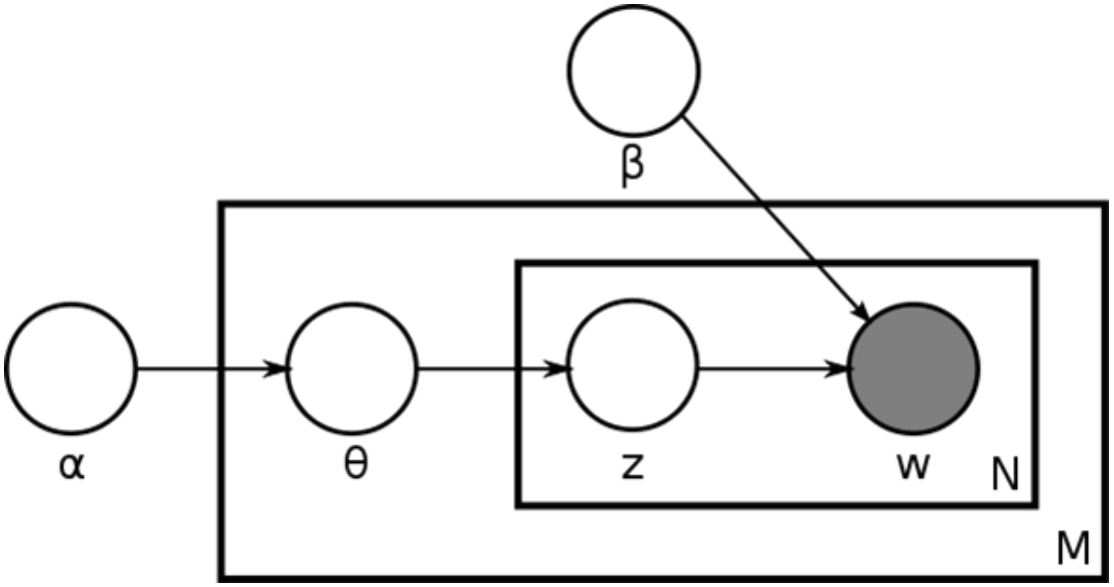
2.4.3.2 pLSA

Probabilistic latent semantic analysis (pLSA), also known as probabilistic latent semantic indexing (pLSI) or aspech model, is a further development from LSA, adding a sounder probabilistic model. While LSA stems from linear algebra and downsizes the occurrence tables via SVD, pLSA is based on a mixture decomposition derived from a latent class model. It models each word in a document as a sample of a mixture model, where the mixture components are multinomial random variables that can be viewed as representation of "topics". The pLSA was tested on 4 dataset: MED, CRAN, CACM, CISI, and has shown improvements on all set from 25.4% to 58.3% precision. This result is also significantly better than LSI, which is not always succeed in showing improvement over raw term matching. This approach is further described in [Hofm99].

2.4.3.3 LDA

Latent Dirichlet Analysis (LDA) is a generative probabilistic model of a corpus. In this approach each topic is modeled as a probability distribution over words, while

each document is modeled as a probability distribution over topics. LDA is similar to pLSA except that it adds Dirichlet prior on the per-document topic distribution. We use LDA to model latent topics of a document collection. The following diagram is a representation of LDA model:



where:

- M: number of documents
- N_d: number of words per document
- α_k: Dirichlet prior
- β_{vk}: unigram probabilities of each latent topic
- θ_d: topic mixture weights
- z: latent topic
- w: word

This graphical representation is called a plate notation. The boxes represent replicates, where the outer box represent documents, and the inner box represent repeated choice of topics and words within a document.

The generative process for each document w in a corpus D can be described in following steps:

1. Choose N which is modeled as a Poisson distribution. Because N is independent of all data generating variables (θ and z), we can safely ignore it's randomness.
2. Choose θ from a Dirichlet distribution $Dir\,(\alpha)$.
3. For each of the N words w_n:

- Choose a topic z_n from a multinomial distribution $Multinomial(\theta)$.
- Choose a word w_n from $p(w_n|z_n;\beta)$, a multinomial probability conditioned on the topic z_n.

Here we can see that there are three levels of variables which represent LDA (d indicates document index and n indicates word index):

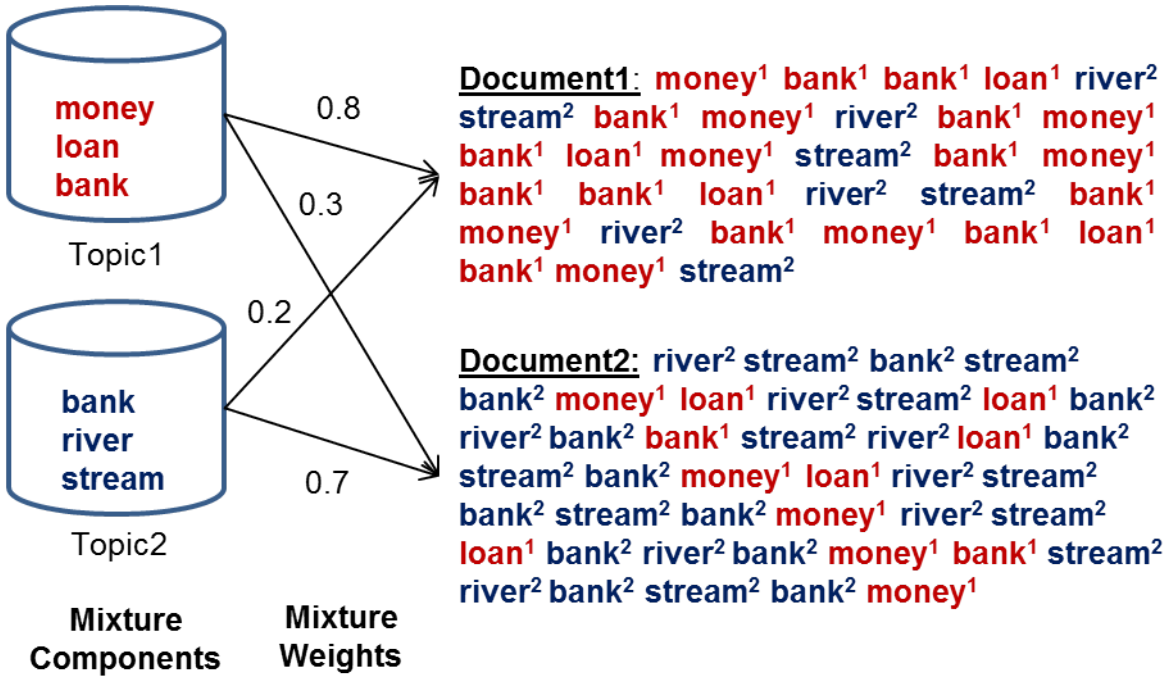
1. α and β: These are corpus level parameters, which means both variables are sampled once for every corpus.
2. θ_d: This is a document level variable which is sampled once for every document.
3. z_{dn} and w_{dn}: Both variables are word level variables which are sampled once for every word.

To bring more understanding to the usefulness of this approach and role of it's parameters, let us consider the following example. Let us define two topics and only five words vocabulary for simplicity.

The corpus parameter β is a $k \times V$ matrix which contains the following probability values:

- $P(money|k_{Topic1}) = 0.3$
- $P(loan|k_{Topic1}) = 0.3$
- $P(bank|k_{Topic1}) = 0.4$
- $P(river|k_{Topic1}) = 0$
- $P(stream|k_{Topic1}) = 0$
- $P(bank|k_{Topic2}) = 0.2$
- $P(river|k_{Topic2}) = 0.2$
- $P(stream|k_{Topic2}) = 0.4$
- $P(money|k_{Topic2}) = 0$
- $P(loan|k_{Topic2}) = 0$

The role of both α and β can be seen in the following:



The mixture components is a Dirichlet distribution with the parameter β which models word probabilities for each topic, while the mixture weights is a Dirichlet distribution with the parameter α. From the example we can see how this approach can differentiate the ambiguous word "bank" from the place we can save money, to a part of land near the river. The topic 1 in this example add probabilities to words related to money and the banking system, while the topic 2 models words related to various part of a river. The mixture weights has the role in mapping the influence of topic 1 and topic 2 to each of the documents.

[BNJL03] describes LDA in further details and how it works.

This thesis is based on [TaSc05] which uses LDA in an unsupervised and Bayesian fashion. Instead of using Mandarin as the language of the training data, in this thesis we want to explore the possibility of improvement LDA can brings with the English data.

3. Analysis & Design

Before conducting the experiment, certain preparation must be done. The data corpus must be cleaned and prepared in a certain way, and the design of the experiment, the training and testing models and parameters, have to be determined.

3.1 Data preparation

This thesis will use English Gigaword corpus for training, which contains 1.8 billion words grouped in 4.2 million documents. For testing purposes, it uses Quaero English data 2010 with 40515 words and 3.7 hours of audio. Before further processing, both data have to be cleaned according to the following steps:

Example of input sentences: * – Less than 20 miles from Singapore's & other skyscrapers is a completely different set of high-rise towers. Green vegetables like bak choi and Chinese cabbage are grown, stacked in greenhouses, and sold at local supermarkets. *

1. Convert encoding to UTF8 (when needed). The data is downloaded from the internet, and webdata usually contains different encodings, two of the most common are UTF8 and latin1 (ISO 8859-1). A cleaning script detected which articles/ documents are not encoded in UTF8, and convert it to UTF8. This is especially relevant for non-ASCII symbols such as various forms of " or non-standard alphabets (umlauts, accented, etc).

2. HTML tags conversion. Data collected from a website often have special HTML-style tags which uses the format similar to &sometext;. These are converted to appropriate characters so that it can be processed further as a normal text. Other unused HTML tags are removed at this stage. Example: *– Less than 20 miles from Singapore's & other skyscrapers is a completely different set of high-rise towers. Green vegetables like bak choi and Chinese cabbage are grown, stacked in greenhouses, and sold at local supermarkets.*

3. Abbreviation & acronym normalisation. Abbreviated words cannot be processed equally along with the unabbreviated version, unless both are normalized. We cannot have both "&" and "and", or "UN" and "United Nations",

because they both mean the same thing and should be evaluated as the same. Example: *– Less than 20 miles from Singapore's **and** other skyscrapers is a completely different set of high-rise towers. Green vegetables like bak choi and Chinese cabbage are grown, stacked in greenhouses, and sold at local supermarkets.*

4. Number normalization. Different types of numbers, dates, range, indexes is normalized to the spoken form. In this way, we have one representation for both "20" and "twenty", and punctuations related to numbers as in "20.34" can be converted to it's word form and will not be lost by the punctuation removal process of the later steps. Example: *– Less than **twenty** miles from Singapore's and other skyscrapers is a completely different set of high-rise towers. Green vegetables like bak choi and Chinese cabbage are grown, stacked in greenhouses, and sold at local supermarkets.*

5. True casing for sentence start. The first word on the beginning of a sentence often contains capitalized letter, which is not capitalized otherwise. This letter is here converted to it's normal form (for example: lower case for verbs, upper case for names, this depends on the language convention). This cleaning step uses ngram-based statistic approach to determine the correct case for the word on each of the sentence start. It ensures the consistency of the word-casing, and eliminates the influence on casing caused by the position of the word (usually the sentence begin). Example:
*– **less** than twenty miles from Singapore's and other skyscrapers is a completely different set of high-rise towers. **green** vegetables like bak choi and Chinese cabbage are grown, stacked in greenhouses, and sold at local supermarkets.*

6. Sentence segmentation. Data collected from websites usually group multiple sentences into paragraphs of sentences. For the purpose of Language Modelling, these paragraphs of text are segmented into one full sentence for each line, so that we know where the sentence start and where it ends. A straightforward solution would be to detect full-stop "." and replace it with a newline. The problem with that is a "." can be used for many other purposes other than ending a sentence. Even when the dots used for abbreviations and number purposes can be assumed to be processed and converted completely, there is still some other usage cases as for example, to write url ("www.apple.com"). This script detect automatically whether a full stop really meant to end the sentence or not, and process it accordingly.
Example:

   - *– less than twenty miles from Singapore's and other skyscrapers is a completely different set of high-rise towers.*

   - *green vegetables like bak choi and Chinese cabbage are grown, stacked in greenhouses, and sold at local supermarkets.*

7. Punctuation and unusable characters removal. Since all punctuations and non-word characters are irrelevant for the calculation being done in this experiment (except certain punctuations which are part of a word as in "Singapore's"), they are removed from the data collection. All punctuations that haven't been read

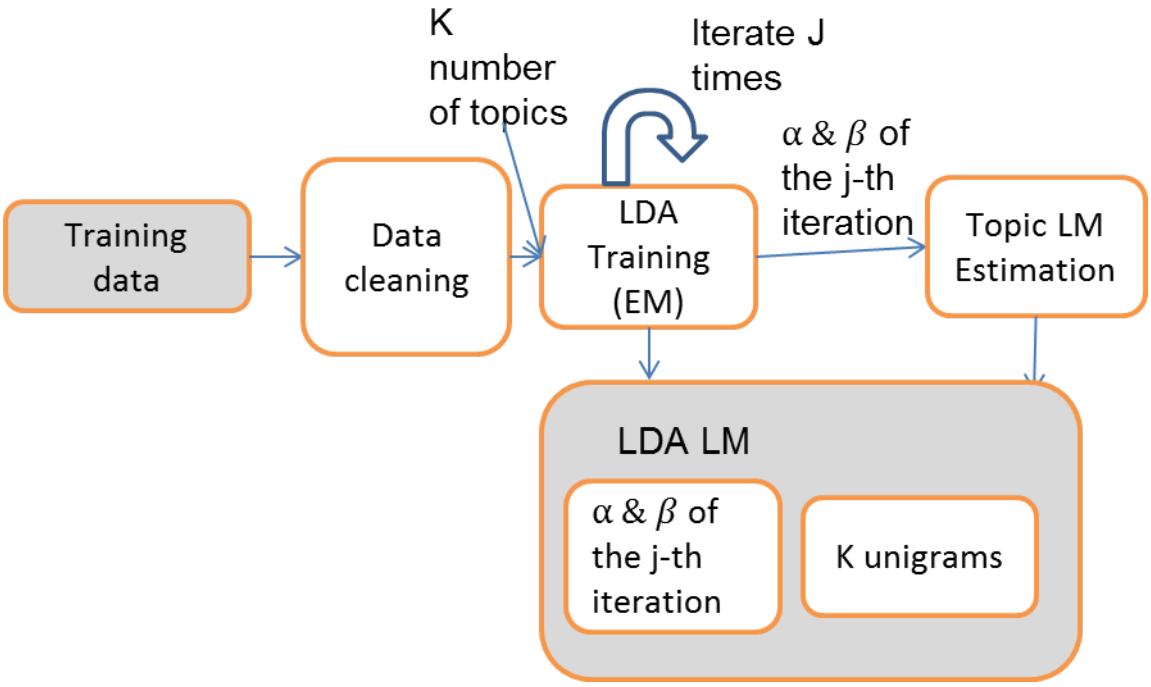
and converted in the previous steps would be gone after this step, so a special check must be done to ensure no words are broken because of this process, e.g. no "Singapore's" should be split into "Singapore s". When the document/ article contains too many unrecognized characters, it might be the case that the downloaded document is not a valid text file. It might be a music mp3 file or a jpeg which was treated as a text file. The cleaning script would also automatically removed such invalid documents.
Example:

- *less than twenty miles from Singapore's and other skyscrapers is a completely different set of high-rise towers*

- *green vegetables like bak choi and Chinese cabbage are grown stacked in greenhouses and sold at local supermarkets*

After the data is cleaned, there is one remaining requirement. The LDA modelling requires the data to be grouped into documents, and marked accordingly. In this case, since the data corpus is a webdata, each webpage is considered as one document, where each document is marked by <text> and </text> to denote document boundaries.

3.2 The training



First we have to determine the number of topics in advance, in the diagram it is being represented by K. To start the training, the cleaned and prepared training data is fed into an LDA training step (which resembles EM step). This step is repeated for J times, in order to estimate the best parameter of the model (represented by α and β). This α and β are being used afterwards for topic LM estimation. The estimation step produces K unigrams, where each unigram list represent one topic. We keep both the α and β of the last (the J-th) iteration, and the K unigrams, and call this an LDA language model.

3.2.1 Variational Bayes-Expectation Maximization algorithm

To acquire the optimized model paramenters (α and β as mentioned in the previous section), we use the variational Bayes-Expectation Maximization (VB-EM) algorithm. As the name suggested, the algorithm consisted of two steps, the Expectation step and the Maximization step. Both steps are described as following: - E-Step:

$$\gamma_k = \alpha_k + \sum_{i=1}^{n} q\left(z_i = k\right)$$

$$q\left(z_i = k\right) \propto \beta_{w_i k} \cdot e^{E_q[log\theta_k]}$$

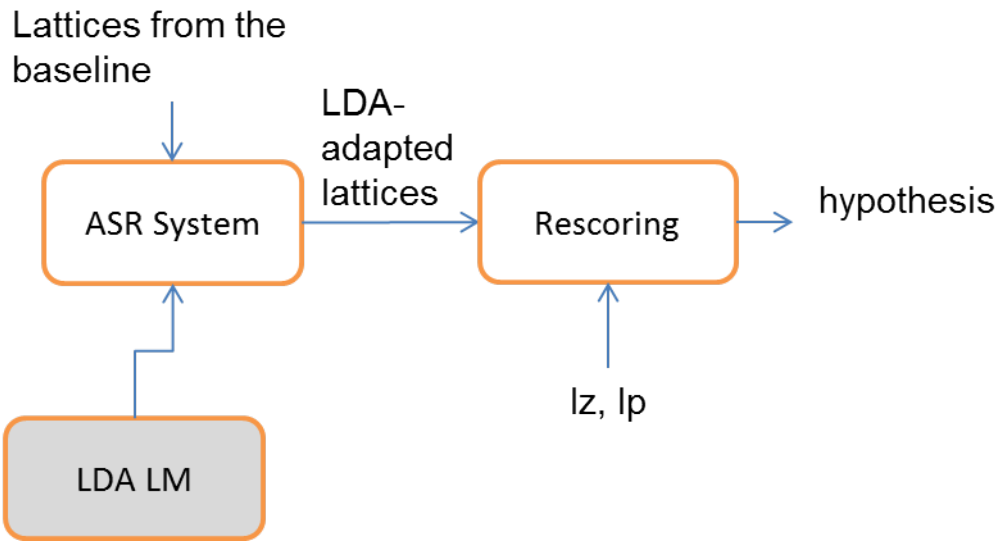
where $E_q\left[log\theta_k\right] = digamma\left(\gamma_k\right) - digamma\left(\sum_{k=1}^{K} \gamma_k\right)$. Both equation are applied iteratively until convergence.

- M-Step:

$$\beta_{vk} \propto \sum_{i=1}^{n} q\left(z_i = k\right) \delta\left(w_i, v\right)$$

where $\delta\left(\cdot\right)$ is the Kronecker Delta function. Newton-Raphson algorithm or gradient ascent procedure is used to determine the parameters of the α_k

3.3 The testing



In this testing step, the LDA LM we acquired from the training step is being applied to adapt the lattices of the baseline system. The adapted lattices are rescored further using certain values of LM weights and word penalty, and the quality of the recognition result is measured. In the following section,

3.3.1 LM Adaptation

The LM adaptation procedure is started by decoding the speech utterance, while store the resulting hypothesis in a buffer. Whenever the buffer exceeds M words, execute the following steps:

1. perform the E-step of VB-EM (see section 3.2.1)
2. update the unigram probability using the Maximum A-Posterior (MAP) likelihood as follows:

$$f(w|h) \approx \int_{\theta} \sum_{k=1}^{K} f(w|z=k) f(z=k|\theta) q(\theta|h)$$

$$\approx \sum_{k=1}^{K} \beta_{wk} \cdot \hat{\theta}_k$$

$$\hat{\theta}_k = \frac{\gamma_k}{\sum_{k=1}^{K} \gamma_k} (k = 1 \cdots K)$$

3. update the Dirichlet prior α as follows:

$$\alpha_k \leftarrow \lambda \cdot \alpha_k + \sum_{i=1}^{M} c_i \cdot q(z_i = k)$$

   where $\lambda \in [0, 1]$ is a scaling factor of the history, and c_i is the confidence score of the i-th word.

4. clear the buffer

The LM adaptation is applied after the speech decoder finished decoding an utterance.

3.3.2 Interpolation with the n-gram model

Because the word order information is ignored in the LDA Model, the so-called b̈ag-of-wordsässumption, we need to combine this model with the existing n-gram LM. The combination of both is implemented by interpolate between both word probabilities using certain weight. The weight used in this study is assigned with 0.8 for the n-gram, and the rest is allocated for the LDA-adapted. The value is chosen based on previous work in [K. K11] which shows that it generally performs well. The interpolation equation is as follows:

$$P(w_i|w_1, w_2, \cdots w_{i-1}) = \mu P_{n-gram}(w_{i-n+1}, \cdots, w_{i-1}) + (1-\mu) f_{LDA}(w|h)$$

where μ is the weight value assigned for the n-gram.

4. Implementation

In this chapter we will discuss various tools and data formats which are important in implementing LDA-based language model adaptation.

4.1 Janus Recognition Toolkit (JRTk)

Janus Recognition Toolkit, also called JRTk or just Janus, is a general purpose speech recognition toolkit developed at the Interactive Systems Labs in Karlsruhe, Germany and Pittsburg, USA. This toolkit is implemented in C, but it is encapsulated in a Tcl/Tk interface. So while the development must be done in C, the normal usage can be done using a Tcl/Tk script. Almost everything of the ASR training and decoding steps are handled by Janus and can be executed by writing certain Janus commands in a Tcl/Tk script. Janus uses Ibis decoder ([SMFW01]), a one-pass decoder which uses the same engine to decode with statistical n-gram language models as well as context free grammars or re-scoring of lattices in a very efficient way.

4.2 LDA library

The LDA library (implemented by [TaSc05]) is a further extension of JRTk toolkit. The existing JRTk functionalities is not changed, but it is extended by a new LSA objects which provides necessary structures and functionalities related to building and applying an LDA model.

4.3 SRILM toolkit

SRILM is a toolkit for building and applying statistical language models (LMs). It consisted of the following components ([SRIL11b]):

- A set of C++ class libraries implementing language models, supporting data stuctures and miscellaneous utility functions.

- A set of executable programs built on top of these libraries to perform standard tasks such as training LMs and testing them on data, tagging or segmenting text, etc.

- A collection of miscellaneous scripts facilitating minor related tasks.

4.4 ARPA back-off file format

The file format used by SRILM for an n-gram model is called ARPAbo, or ARPA back-off format. It was developed by Doug Paul at MIT Lincoln Labs for research sponsored by the U.S. Department of Defense Advanced Research Project Agency (ARPA) ([SRIL11a]). This file format will be used in the LDA-based training to display unigram probability values for each topics.

The ARPAbo format is started with a header, marked by the keyword
data
, and it is followed directly with the number of n-grams for each n. Afterwards each n-grams are listed, one n-gram sequence per line, grouped into sections for each n, where each section starts with the logarithmic probability of the n-gram, the n-gram word sequence, and optionally the back-off weight for the n-gram. The keyword
end
marking the end of the file.

\data\

ngram 1=n1

ngram 2=n2

...

ngram N=nN

\1-grams:

p w [bow]

...

\2-grams:

p w1 w2 [bow]

...

\N-grams:

p w1 ... wN

...

\end\

4.5 FSM toolkit

AT&T FSM library [Rese] provides algorithms and representations for phonetic, lexical, and language-modeling components of large-vocabulary speech recognition systems. In this thesis it is used to rescore the lattices produced by the baseline system using the LDA probability.

The following binary executables from FSM library used in this thesis are described:

- fsmcompile: takes input file representing an FSM and sends to standard output its binary encoding. The input should be the textual representation of an FSM.
- fsmbestpath: returns the lowest-cost path from the start state of the input FSM to a final state. The path is encoded as a (single path) FSM.
- fsmprint: prints the input FSM on standard output using same textual format as what fsmcompile accepts as input.

4.6 Scoring toolkit

To align the hypothesis to the reference text and produce WER, the LNEtools is used. LNEtools is a toolkit used for Quaero evaluations which are developed by the LNE (Laboratoire1www.quaero.org National de M'etrologie et d'Essais2). It not only returns a WER for a hypothesis, but it also produces a thorough analysis on the result. It produces a sentence by sentence alignment, where we can see in which sentences or cases the recognition failed badly, and it also provides confusion pairs, WER for each speakers, and many other useful data that can be used to improve the system.

5. Evaluation

In this chapter, the results from each of the experiment will be evaluated and analyzed.

5.1 Baseline System

The baseline system used is an English ASR developed for the Quaero 2010 English evaluation. At first we build two systems with different acoustic front-ends. The first acoustic model is trained based on Mel-frequency Cepstral Coefficients (MFCC) obtained from a fast Fourier Transform, and the second one based on warped minimum variance distortionless response (MVDR) [WoMW03]. Warped MVDR spectral envelope used in the second front-end provides the properties of Mel-filterbank, therefore it doesn't require any filterbank. Both front-ends provided a feature every 10 ms for training, but during decoding it was changed to 8ms after the first stage. Both apply vocal tract length normalization (VTLN) [ZhWe97], which is done in the linear domain for MFCC and warped frequency domain for MVDR. The MFCC uses 13 cepstral coefficients, the MVDR uses 15. Mean and variance of the cepstral coefficients were normalized on a per-utterance basis. For both MFCC and MVDR, the frame in the current position and seven adjacent frames were stacked together into one single feature vector, bringing the total into 15 frames for each feature vector. This resulting feature vector were then reduced to 42 dimensions using linear discriminant analysis (LDA).

Acoustic models are semi-continuous quinphone systems using 16000 distributions over 4000 codebooks. They were trained using incremental splitting of Gaussians training, and then followed by 2 iterations of Viterbi training. Constraint MLLR (cMLLR) speaker adaptive training [Gale97] was applied, and then Maximum Mutual Information Estimation (MMIE) training to improve the models further.

The decoding was performed in two stages. The first stages produced word lattices for each front-ends. It is then combined via confusion network combination (CNC). The acoustic models of the second stage were adapted on the combined output from the first stage using Maximum Likelihood Linear Regression (MLLR) [LeWo95], VTLN [ZhWe97], and cMLLR [Gale97]. We take the result of the MFCC of the

second stage to be the baseline of this tesis. The baseline language model was built using modified Kneser-Ney smoothed 4-gram language model, which was pruned for efficiency. Further details for the steps to produce the baseline system are described in [K. K10].

The best result for the baseline system is obtained using LM weights = 36 and word penalty = -4, which resulted in 31.99% WER. This WER value and the lattices producing it are used as a baseline for any experiments done this thesis.

5.2 LDA-adapted System

The LDA-model resulted from the previous training cycles, is being used to rescore the baseline lattices. The resulting LDA-adapted lattices are afterwards used to produce hypothesis, which are evaluated and measured against the baseline by looking at it's WER.

5.3 Experiments

Because of the existence of uncertain parameters, each of the parameter has to be tested using multiple possible values and evaluated. There are three experiment which test three categories of parameters:

1. Number of iteration of LDA Training (the EM step). This determines the quality of α and β (the LDA model parameters). The values being tested are: 10, 15, 20, 25, 30, 35, and 40 iterations

2. Number of latent topics. This determines the number of topics which is used to model the document. Each topic is a mixture of word probabilities. Each document is modeled as a mixture of topic probabilities. The values being tested are: 25, 50, 100, 150, 200 topics

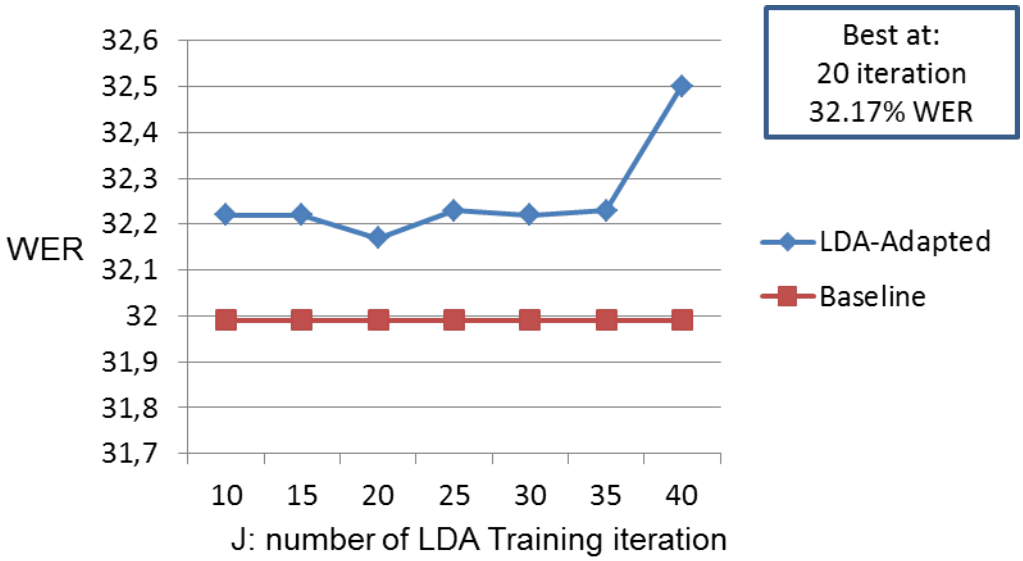
3. LM weights and word penalty. This parameters are used in mixing the language model and acoustic model. To test this we use a matrix using LM weights range: 28, 32, 36, 40, 44, and word penalty range: -8, -6, -4, -2

For each experiment, only the parameters tested are varied, while the rest of the parameters are held constant. The third test was done after the first two tests has been completed, so that the combination of the best values from the first two tests can be tested using various LM weights and word penalty in order to get a better result.

5.4 Results

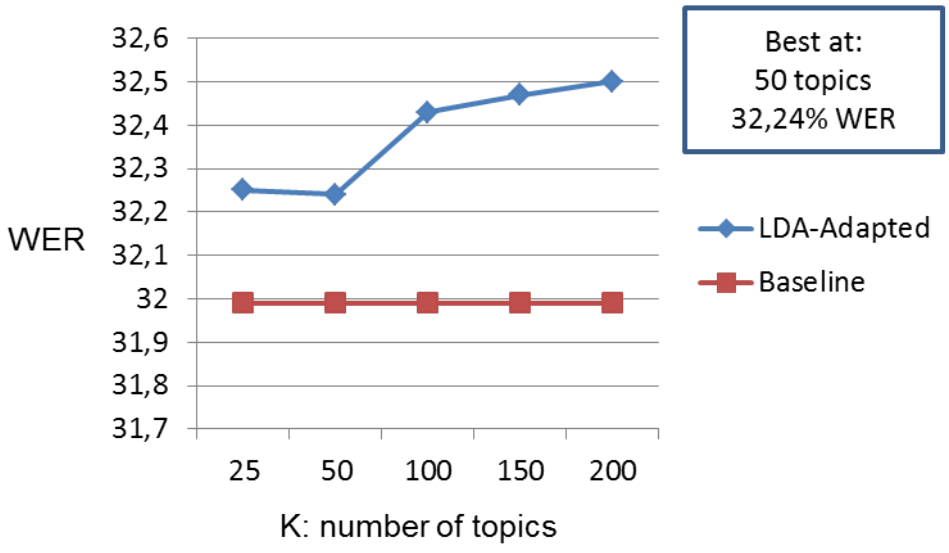
The following are the word error rate (WER) recognition result of each of the test. The WER of the LDA-based ASR are shown alongside the WER of the baseline system, so it can be seen easily whether the new system improve the accuracy of the ASR.

5.4.1 Number of Iteration



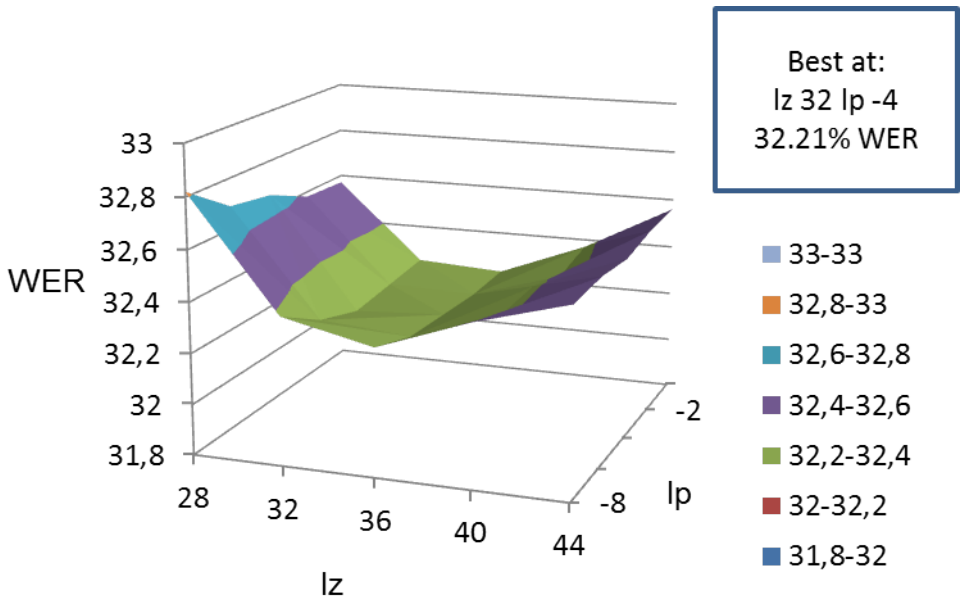
As we can see from the result, unfortunately there is no improvement from the LDA-based system. The baseline produced 31.99% WER, but none of the parameter values shows any improvement. For the first 35 iteration it returns roughly the same result, with around 0.2% worse on the WER. As soon as it reaches 40 iteration the WER spikes into much worse values. Other parameters held constant are the number of topics (200 topics), LM weights = 36, and word penalty = -4.

5.4.2 Number of Topics



For the second test it can be seen clearly that for the number of topics larger than 50 the WER got a lot worse. The best number of topics is found at 50, which correspond to the original study in [TaSc05]. However, the similar result is produced as the first test, where there is no WER improvement on the baseline. Other parameters used are number of iteration (held at 40 iterations), LM weights = 36, and word penalty = -4.

5.4.3 LM weights and word penalty



The third test simply try out different LM weights (represented by the variable lz on the diagram) and word penalty (represented by the variable lp). For the number of iteration it chooses the best parameter tested in the first test, which is 20 iterations. The number of topics is set at 50 topics, which is the best number of topics resulted from the second test.

The best WER value is found at LM weights = 32 and word penalty = -4 with 32.21%. Similar to the previous tests there is no improvement on the baseline. Unexpectedly it is even slightly worse compared to the best WER in the first test 32.17% (using 20 iteration, 200 topics).

5.4.4 Further analysis

Other than WER, a deeper look into examples of hypothesized utterances produced by LDA-adapted system does show small difference to the base line. The following are parts of the decoding result from both with and without LDA-adaptation as an example:

The reference transcript:
no one has ever claimed responsibility for Jean Mcconville's murder but there is speculation in Northern Ireland that a new book by the journalist Ed Moloney may reveal the killer's identity he interviewed the IRA leader Brendon Hughes who died in two thousand and

Produced by the baseline system:
no one has ever claimed responsibility for **GENE MCCALL MOVES** murder but there is speculation in northern ireland that a new book by the journalist **AND MALONE** may reveal the killer's identity he **INTERVENED** the IRA leader **BRENDON** Hughes who died in two thousand and

Produced by LDA-adapted system:
no one has ever claimed responsibility for **GENE BUT CONGOLESE** murder but there is speculation in northern ireland that a new book by the journalist **AND**

MALONE may reveal the killer's identity he **INTERVENED** the IRA leader **BRENDON** Hughes who died in two thousand and

As we can see there is small differences between both. These differences can be attributed to extra topic information we get from LDA adaptation. When we look on the 1st sentence, the word "Congolese" might have a closer topic to the word "murder" rather than "moves". It unfortunately did not help the WER since the new words predicted by the LDA-adapted system is still not the correct words as written in the references. There are two possibilities that can cause this. It could be the case that the LDA-adaptation is not good enough to help the system predict the correct words because either the parameter or the model itself is not optimal. "Congolese" is still not the correct word. Or, there is also another possibility that the baseline system is already very good, so that words boosted by the LDA-adaptation method (IRA, leader, murder, killer's, etc) are already predicted correctly even without any adaptation, hence the lack of improvement on the WER.

6. Summary

In this work, various LDA parameters has been tested and each LDA model has been applied to the baseline lattice of the English Quaero data 2010. It has been tested using various number of training iteration and the number of topics used, and afterwards the LM weights and word penalty have also been optimized. However, by looking at the WER recognition results produced by the LDA-adapted lattices, it can be seen that the current approach has not improved the baseline system's WER.

6.1 Further works

The scope and the number of parameters tested should be expanded further. This might make sure we get the global minimum value for the WER instead of the local minima. The quality of the training corpus can also be improved by applying better vocabulary and better text preprocessing, both the text cleaning and the document tagging.

There is a problem on the FSM lattice format used in the implementation of the lattice rescoring. It requires a huge amount of space (around 400 GB for each test, uncompressed), and it accordingly spent too much on the calculating power. Better implementation using other library or better lattice optimization have a potential to immensely decrease the time needed for completing the decoding process, which would made possible many more tests being done using more parameters and/or values.

Bibliography

[BNJL03] D. M. Blei, A. Y. Ng, M. I. Jordan und J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research* Band 3, 2003, S. 2003.

[ClRo97] P. Clarkson und A. J. Robinson. Language Model Adaptation Using Mixtures And An Exponentially Decaying Cache. In *In Proceedings of ICASSP-97*, 1997, S. 799–802.

[DDFL+90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer und R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6), 1990, S. 391–407.

[Gale97] M. Gales. Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition. *Computer Speech and Language* Band 12, 1997, S. 75–98.

[Hofm99] T. Hofmann. Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI'99*, 1999, S. 289–296.

[JeBM75] F. Jelinek, L. R. Bahl und R. L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory* Band 21, 1975, S. 250–256.

[K. K10] S. S. K. Kilgour, J. Niehues. Quaero Speech-to-Text and Text Translation Evaluation Systems. 2010.

[K. K11] S. S. A. W. K. Kilgour, F. Kraft. Multi Domain Language Model Adaptation using Explicit Semantic Analysis. 2011.

[Katz87] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1987, S. 400–401.

[KnNe95] R. Kneser und H. Ney. Improved backing-off for n-gram language modeling. In *In Acoustics, Speech, and Signal Processing*, 1995.

[LaRR93] R. Lau, R. Rosenfeld und S. Roukos. Trigger-based language models: a maximum entropy approach. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 1993, S. 45–48.

[LeWo95] C. J. Leggetter und P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models, 1995.

[Rese] A. L. Research. FSM Library.

[SMFW01] H. Soltau, F. Metze, C. Fügen und A. Waibel. A One-Pass Decoder Based on Polymorphic Linguistic Context Assignment, 2001.

[SRIL11a] SRILM. File format for ARPA backoff N-gram models, 2011.

[SRIL11b] SRILM. The SRI Language Modeling Toolkit, 2011.

[TaSc05] Y.-C. Tam und T. Schultz. Dynamic Language Model Adaptation using Variational Bayes Inference. 2005.

[WiBe89] I. H. Witten und T. C. Bell. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. In *IEEE Transactions on Information Theory*, Band 37. IEEE, 1989, S. 1085–1094.

[WoMW03] M. Woelfel, J. W. McDonough und A. Waibel. Minimum variance distortionless response on a warped frequency scale. In *INTERSPEECH*. ISCA, 2003.

[ZhWe97] P. Zhan und M. Westphal. Speaker Normalization Based On Frequency Warping, 1997.