Karlsruhe Institute of Technology

Language Model Adaptation using Temporal Information

Diplomarbeit

of

Karl Karnadi

at the Institut for Anthropomatics

of the Faculty of Informatics

Start: September 1, 2013

End: January 13, 2014

Advisors

Kevin Kilgour

Sebastian Stueker

Contents

1. Introduction

Broadcast news has brought us a window of information into events happened in the world. In the past, newspapers and magazines has a role to provide the news collected after certain period of time (weekly, daily). In the recent decades, news has been increasingly provided via the internet and TV. In contrary to newspapers, news can be written and published instantly in hours or even minutes after the event happened. Even with this almost-real-time news publishing, broadcast news have certain patterns of regularities in the form of trending topics.

After a big event happens in the world, news channels and websites broadcasted news and discussions about the event repeatedly for certain period of time. An event like a natural disaster in the Philippines Typhoon Haiyan, for example, would trigger a flurry of news articles written about the disaster, the victims, the cost, and the recovery. All of them would stay in the news for at least several weeks.

The transcription of broadcast news has been done both manually and automatically in the past. This transcript is very helpful because it can be a source of translation into other languages, which made news exchanges between countries and languages easier, and it can also help those who have hearing disabilities. An automatic speech recognition (ASR) system is used to reduce both time and cost needed for the transcription process.

In an ASR system, the accuracy of the system depended heavily on the distance between the training data and the data used to evaluate the system. When the evaluation data contains articles about a natural disaster, it would be helpful when the training data also contains texts mentioning the same natural disaster. The phenomenon of how trending topics stayed for certain period of time in broadcast news mentioned above, can be exploited to improve the quality of an ASR system.

An ASR system automatically converted an audio of a spoken words into transcribed text. It has different parts that work together to produce result. It can be modelled with the following equation:

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

To find best hypothesized words $\hat{W}$, a sequence of words is chosen which best match the acoustic input X. It was implemented by find maximum conditional probability of word sequence based on the acoustic information. Applying Bayesian equation and further simplification resulted in the equation above, with $P\left(X|W\right)$ represent Acoustic Modelling (AM) and $P\left(W\right)$ is represent Language Modelling (LM). In this thesis, the adaptation of the LM based on the trending news with it's temporal information was explored.

1.1 Objectives

The purpose of this work is to improve the recognition result of an English speech recognition system which already uses large language model, by using a collection of news articles with it's temporal information.

1.2 Outlines

In chapter 2 the Language Modelling and the phenomenon of trending news will me introduced and described. The same chapter will also discuss techniques and related works used in this thesis. Chapter 3 explains data preparation, the training and testing schemes designed for the experiments. Chapter 4 explores experimental setup and tools used in this work. The chapter 5 presents results with different test scenarios, and the chapter 6 provide a summary and suggestions for future works.

2. Overview

This chapter described the theories behind language modelling, and demonstrated the influence of trending topics for broadcast news. The latter section presented related works and techniques used for this thesis.

2.1 Language model

A language model is represented by a probability distribution $P\left(W\right)$. This represented the probability of a sequence of words occur in the data. The probability of the English sentence "good morning", for example, would be much higher than the probability of "good mourn ink". Similar acoustic sound produced ambiguities, which can only be solved by the language model.

The straightforward approach to language model uses the following equation:

$$P\left(W\right) = P\left(W_1, W_2, \cdots, W_n\right) = P\left(W_1\right) P\left(W_2 | W_1\right) \cdots P\left(W_n | W_1, W_2, \cdots, W_{n-1}\right) \tag{2.1}$$

This, however, would require extremely large amount of data to estimate the probability value accurately. The most often used solution is n-gram technique, which approximated the probabilities by only considering the most recently occured $(n-1)$ words.

2.2 N-gram

The most widely used statistical-based approach to build a language model is n-gram. An n-gram is a sequence of n consecutive words inside a text corpus. The model predicts conditional probability of a word W_i based on the history of the occurrence of words on the text corpus. For the basic model, previous $n-1$ words $W_{i-1}, W_{i-2}, ..., W_{i-(n-1)}$ is used as the history.

$$P\left(W_i|h_i\right) \approx P\left(W_i|W_{i-n+1}...W_{i-1}\right) \tag{2.2}$$

N-gram has common alternative names for certain number of n. For two words it is called a bigram, and for three words it is trigram. To estimate the probability values accurately, large amount of text is used. This collection of text is called a training corpus. The n-gram can be estimated by counting the frequencies of the of the word pair $C\left(W_{i-n+1}...W_i\right)$ and $C\left(W_{i-n+1}...W_{i-1}\right)$ and compare them as follows:

$$\hat{P}\left(W_i|W_{i-n+1}...W_{i-1}\right) = \frac{C\left(W_{i-n+1}...W_i\right)}{C\left(W_{i-n+1}...W_{i-1}\right)} \tag{2.3}$$

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

$$P\left(John\ read\ a\ book\right) = P\left(John|\text{<s>}\right)P\left(read|John\right)P\left(a|read\right)P\left(book|a\right)P\left(\text{</s>}|book\right) \tag{2.4}$$

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:

$$P\left(read|John\right) = \frac{C\left(John\ read\right)}{C\left(John\right)} \tag{2.5}$$

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

Just as more complex model requires more data, a more complex n-gram requires corpus with larger number of sentences to estimate the probability values accurately. Apart from the size of the corpus, the training corpus should have the same domain, or even better, a similar topic compared to the evaluation data, in order to best model it.

The advantage of the n-gram model is on it's flexibility in modeling diverse usage of sentences. In contrary to rule-based or grammar-based models, It can capture spontaneous speech, which are often contains grammatical errors. Some important short distance grammatical features can also be captured by the n-gram model. However, there are several problems in implementing n-gram which are addressed and solved in the following subsections.

2.2.1 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 data 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:

$$P_{smooth}\left(W_i \mid W_{i-n+1}^{i-1}\right) = \begin{cases} \alpha\left(W_i \mid W_{i-n+1}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) > 0 \\ \gamma\left(W_i \mid W_{i-n+1}^{i-1}\right) P_{smooth}\left(W_i \mid W_{i-n+2}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) = 0 \end{cases} \tag{2.6}$$

$$\text{where } w_p^q = w_p \ldots w_q \tag{2.7}$$

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.

$$P_{smooth}\left(W_i \mid W_{i-n+1}^{i-1}\right) = \begin{cases} \alpha\left(W_i \mid W_{i-n+1}^{i-1}\right) & \\ +\gamma\left(W_i \mid W_{i-n+1}^{i-1}\right) P_{smooth}\left(W_i \mid W_{i-n+2}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) > 0 \\ \gamma\left(W_i \mid W_{i-n+1}^{i-1}\right) P_{smooth}\left(W_i \mid W_{i-n+2}^{i-1}\right) & \text{if } c\left(W_{i-n+1}^{i}\right) = 0 \end{cases} \tag{2.8}$$

$$\text{where } w_p^q = w_p ... w_q \tag{2.9}$$

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

2.3 LM evaluation

Language model quality can be measured independently using Perplexity. To judge the quality of the whole system however, Word Error Rate (WER) must be measured. Both measurements are used in this thesis to demonstrate the performances of different systems.

2.3.1 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:

$$P(T) = \prod_{i=1}^{N} P(W_i) \tag{2.10}$$

The cross-entropy of the model is defined as:

$$H_p(T) = -\frac{1}{W_T} log_2 P(T) \tag{2.11}$$

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

The perplexity is calculated by the following equation:

$$\text{ppl} = 2^{H_p(T)} \tag{2.12}$$

Perplexity indicated how good an LM modeled a sequence of words in a body of text. Lower perplexity indicated a better modeling, which generally provided better recognition results. The perplexity also indicated the distance between training and evaluation data, which is very useful for our purposes.

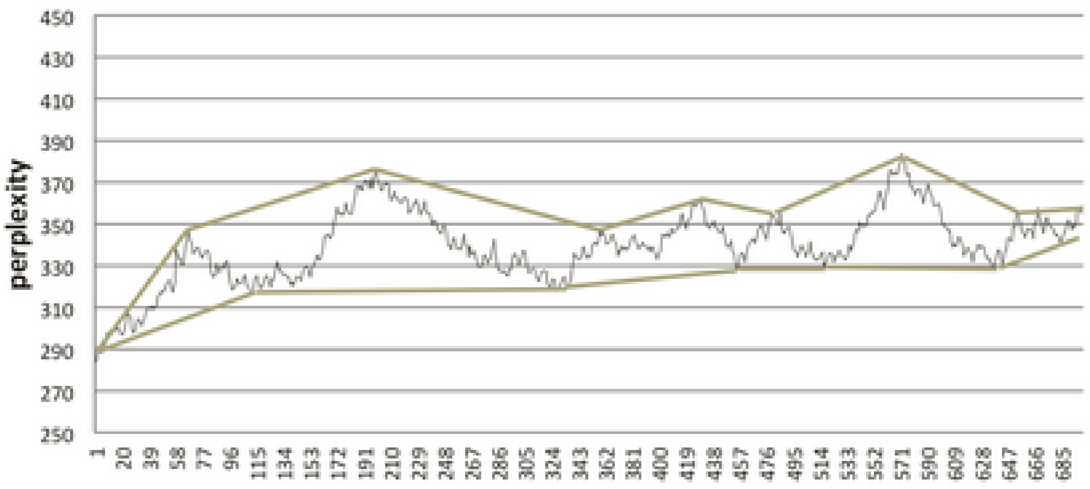
2.3.2 Word Error Rate

To measure the quality of both acoustic and language model which contributed toward the recognition accuracy, the quality of the hypothesized text itself must be measured against a reference text. The Word Error Rate (WER) is calculated by measuring the minimal edit distance between the hypothesis and the reference sentence, which is the minimal numbers of substitutions s, insertions i, and deletions d required to transform the hypothesis into the reference. The WER is calculated as follows:

$$WER = \frac{s+i+d}{n} \times 100\% \tag{2.13}$$

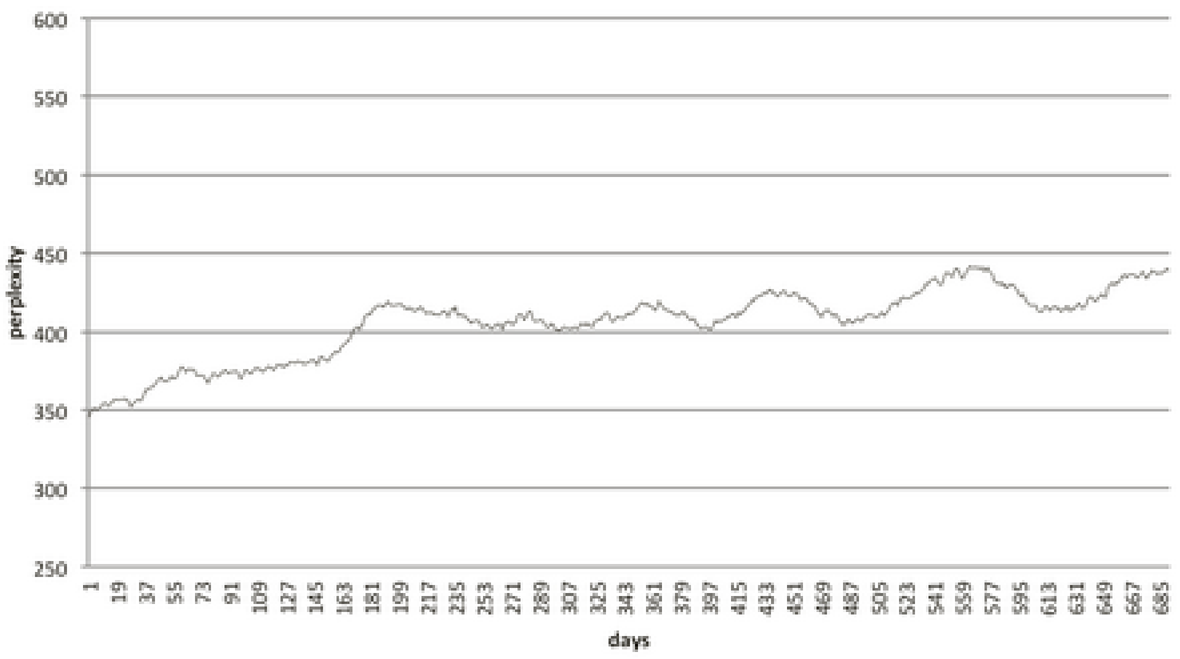
where n is the number of words in the reference sentence.

2.4 Trending News



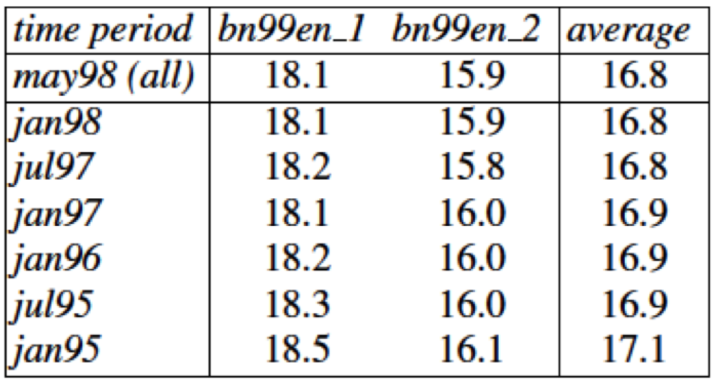
On the above perplexity chart, we can observe the hills and valleys of the perplexities of the training data, measured against a CNN news published in March 11, 2010. The x-axis represented the days which were used as a starting point of a 30-days window of collected text, used for training an LM. For example the point 1 would indicate a perplexity of an LM trained using articles collected between March 10 and 30 days before March 10.

The low perplexities indicated window of times when certain topics mentioned in the evaluation data, for example Greek bailout, became trending news in news media. A new development on certain news topic, for example a new bailout package for Greece, would cause a new increase of the discussion regarding the topic on the news, which would lower the perplexity on that particular day. Beside this phenomenon, there is a long-term trend that the perplexity got worse the further away the training data's date from the evaluation date.



Using nine different shows and make a weighted average based on the number of sentences for each show, the perplexity chart seems to be much smoother. Here the trend of worsening perplexity can be seen much more clearly.

The influence of the temporal distance between training and evaluation data can also be seen in terms of WER, which has been observed in other works such as [CLGA04]



time period	*bn99en_1*	*bn99en_2*	*average*
may98 (all)	18.1	15.9	16.8
jan98	18.1	15.9	16.8
jul97	18.2	15.8	16.8
jan97	18.1	16.0	16.9
jan96	18.2	16.0	16.9
jul95	18.3	16.0	16.9
jan95	18.5	16.1	17.1

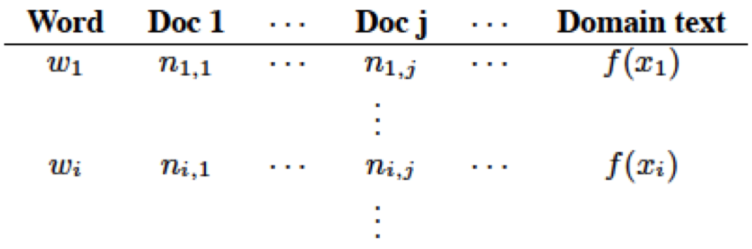
On the table listed the WER of different systems trained using training data from different dates (from July 1995 to January 1998), measured based on the evaluation data from the year 1999. It can be seen that the system with the closest temporal distance between training and evaluation dates returned best performance. The larger the temporal distance is, the worst the performance gets.

2.5 Related works

In this section several related works will be discussed.

2.5.1 Vocabulary Selection

With the inclusion of new text for building a language model, there was a new question of how to incorporate newly found words. Adding all new words directly would resulted in the large size of vocabulary, which would worsened the performance of the system. Using a small size vocabulary, in other hand, would prevented us from exploiting optimally the new information we got from the new data.



Word	Doc 1	$\cdots$	Doc j	$\cdots$	Domain text
w_1	$n_{1,1}$	$\cdots$	$n_{1,j}$	$\cdots$	$f(x_1)$
			$\vdots$		
w_i	$n_{i,1}$	$\cdots$	$n_{i,j}$	$\cdots$	$f(x_i)$
			$\vdots$		

A technique of vocabulary selection was developed by Venkataraman and Wang in [VeWa03] to solve the problem. A true counts x_i for word w_i is estimated based on different counts $n_{i,j}$ from a number of documents.

$$x_i = \Phi_i\left(n_{i,1}, \cdots, n_{i,m}\right) \tag{2.14}$$

Modeling Φ_i as linear functions of the $n_{i,j}$ independent of the particular words w_i, we can write the function as

$$\Phi_i\left(n_{i,1}, \cdots, n_{i,m}\right) = \sum_j \lambda_j n_{ij} \tag{2.15}$$

which transformed the problem into learning λ_j. It was found that the Maximum-Likelihood estimation is the best method in estimating them. The calculation was done iteratively using the following procedure

$$\lambda_j \leftarrow 1/m \tag{2.16}$$

$$\lambda'_j \leftarrow \frac{\lambda_j \prod_{i=1}^{|V|} \mathrm{P}\left(w_i|j\right)^{C(w_i)}}{\sum_k \lambda_k \prod_{i=1}^{|V|} \mathrm{P}\left(w_i|k\right)^{C(w_i)}} \tag{2.17}$$

$$\delta \leftarrow \lambda'_j - \lambda_j \tag{2.18}$$

$$\lambda_j \leftarrow \lambda'_j \tag{2.19}$$

Repeat from (2.17) if $\delta >$ some threshold

As a result of the selection we get a list of words reverse-sorted based on it's estimated count value. Since the top words are the most important words to incorporate to the training vocabulary, we can incorporate these words in a number of ways. We will discuss this further in the next chapters.

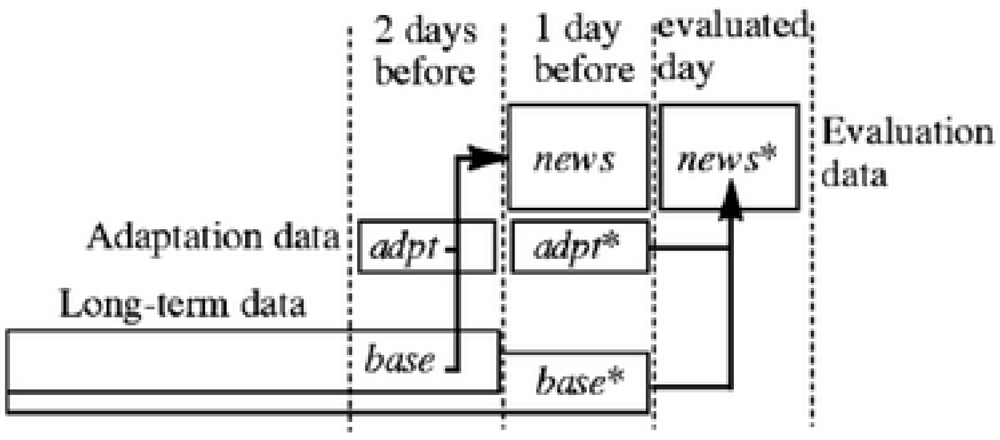
2.5.2 Time Dependent Language Model

[KOIA98] has a similar idea which was further explored by this thesis. It achieved 4.4% reduction in perplexity and 0.7% improvement in word error rate. The data was divided into two categories: long-term news, and short-term latest news.

- Long-term news is the large data that is commonly used for the training. For this purpose, they used articles collected during five years period before the evaluation date.
- Short-term latest news is the much smaller data collected from one or several days before the evaluation date.

The novel idea is to adapt the long-term news using the short-term, since the short-term data is better match the evaluation data in terms of topics or news event, and in the same time the long term data provide more accurate n-grams.

2.5.2.1 Pseudo-weight



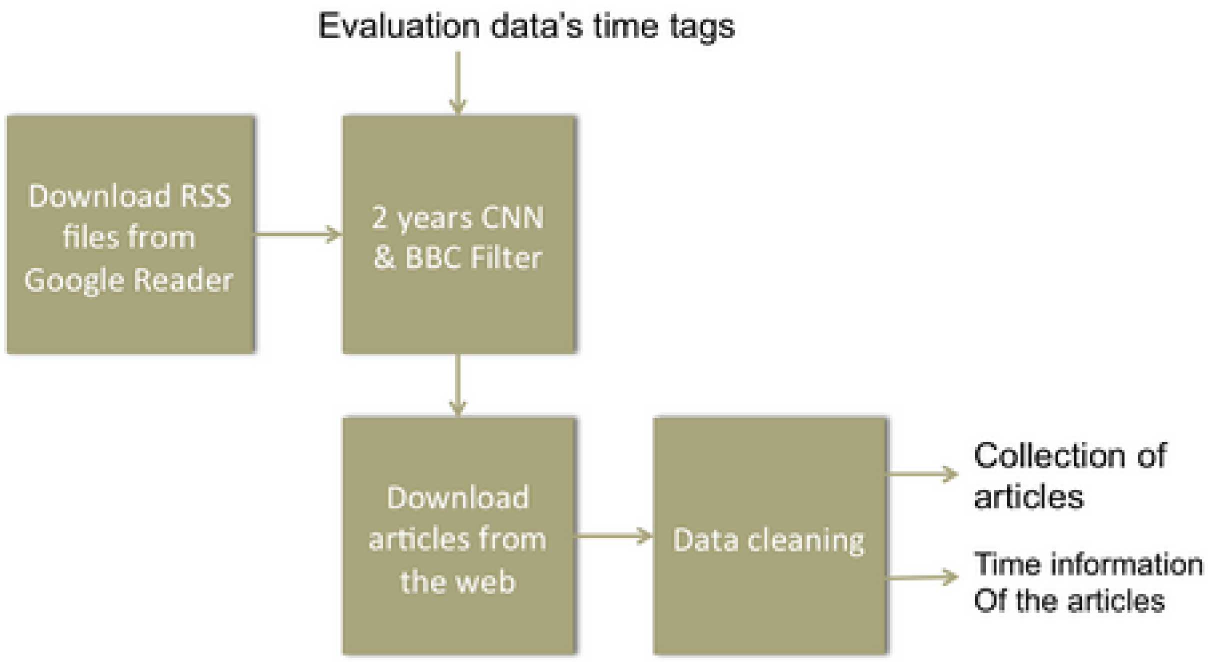
The small language model using articles from 1 day before the evaluation date, was used for the adaptation by being mixed with the baseline language model. The mixing weight is acquired by using an EM algorithm, based on pseudo-evaluation data, which has the evaluation date, shifted 1 day toward the past. Both adaptation LM and baseline LM used to approximate the mixing weight were also shifted 1 day to the past. The resulting mixing weight then applied to the actual adaptation and baseline LM to produce adapted LM.

In this thesis the pseudo-weight approach will be used to acquire mixing weights.

3. Analysis & Design

Before the experiment, the training data would need to be collected and cleaned. The adaptation would also need to be planned and prepared.

3.1 Data Collection


Evaluation data's time tags
Download RSS files from Google Reader
2 years CNN & BBC Filter
Download articles from the web
Data cleaning
Collection of articles
Time information Of the articles

For the purpose of adaptation in this thesis, a collection of articles is needed, along with the publishing date information for each of the article. Such information is provided by the RSS file. The Rich Site Summary (RSS) file is a short file published by most major news website contains the latest articles from that website. The problem with the normal RSS file is that it only provided 50-100 (or at most thousands) of the most recent articles, which wasn't enough for our experiment.

Google Reader is one of the RSS providers which has been crawling and collecting RSS files since many years ago. So the first steps of the data collection is to download this complete RSS collection and then to filter and select only two years before each evaluation shows, and only selecting RSS files published by the CNN and BBC. These list of articles then were downloaded, so that in the end we get collection of articles and it's temporal information.

3.1.1 Data Cleaning

The website articles downloaded were initially in html format. The relevant data is the news texts, and they have to be separated from the menu texts, copyrights, advertisements, etc, and have to be presented in a standard format, which also made easier further processing into n-gram model according to various experiments that will be executed through out this thesis. The following are the stages of data cleaning and examples of each stages.

3.1.1.1 Uncleaned downloaded article

The text as downloaded from the web.

Example: *<a href=...>Home</a><p> – Less than 20 miles from Singapore's &amp; other skyscrapers is a completely different set of high-rise towers. Green vegetables like bak choi &amp; Chinese cabbage are grown, stacked in greenhouses, &amp; sold at local supermarkets.</p>*

3.1.1.2 Paragraph selection

Only selecting texts inside paragraph tags and ignore the rest. By doing this we exclude most of irrelevant texts (menu titles, advertisement, etc).

Example: – Less than 20 miles from Singapore's &amp; 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.

3.1.1.3 UTF8 encoding

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).

3.1.1.4 HTML tags normalization

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 unrecognized 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.*

3.1.1.5 Abbreviation & acronym normalization

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.*

3.1.1.6 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.*

3.1.1.7 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.*

3.1.1.8 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.*

3.1.1.9 Punctuation and unused 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. “Singapore's” should not 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*

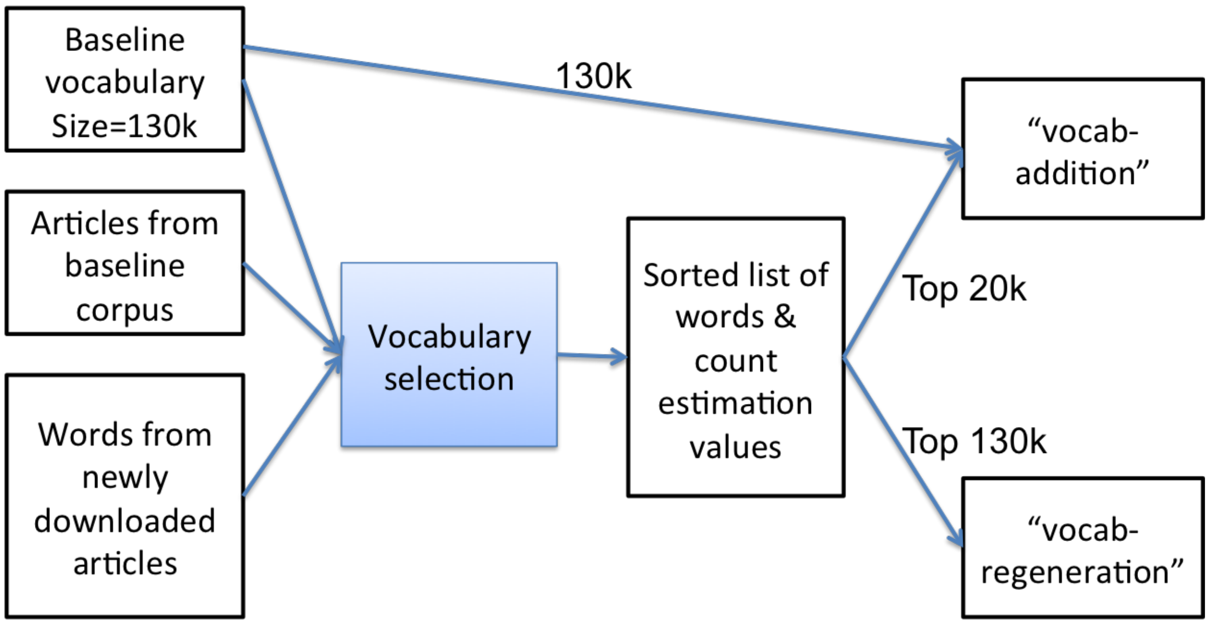
- *green vegetables like bak choi and Chinese cabbage are grown stacked in greenhouses and sold at local supermarkets*

Additional cleaning can be done afterwards, to clean special cases that haven't been handled by the existing cleaning steps, but occur quite often. With the limited website source used in this thesis (CNN and BBC), the format of the article or the website is much more predictable, so the cleaning can be done much more thorough and handling special cases won't risk corrupting existing cleaned sentences.

3.2 Adaptation Design

There are two parts of adaptation: the vocabulary adaptation, and the adaptation to the probability values in the LM itself. Both parts are described in this section.

3.2.1 Vocabulary selection


Baseline vocabulary Size=130k
Articles from baseline corpus
Words from newly downloaded articles
Vocabulary selection
Sorted list of words & count estimation values
130k
Top 20k
Top 130k
"vocab-addition"
"vocab-regeneration"

Every time a LM was built, a set of limited vocabulary is needed. A vocabulary selection process is done before any adaptation process started. Vocabulary selection algorithm and equations has been described in 2.5.1. The selection resulted in a list of words, which was sorted by a value indicating each of the words' importance. The words on the top are the most important words to be used in the training vocabulary, and the words on the bottom are the least important.

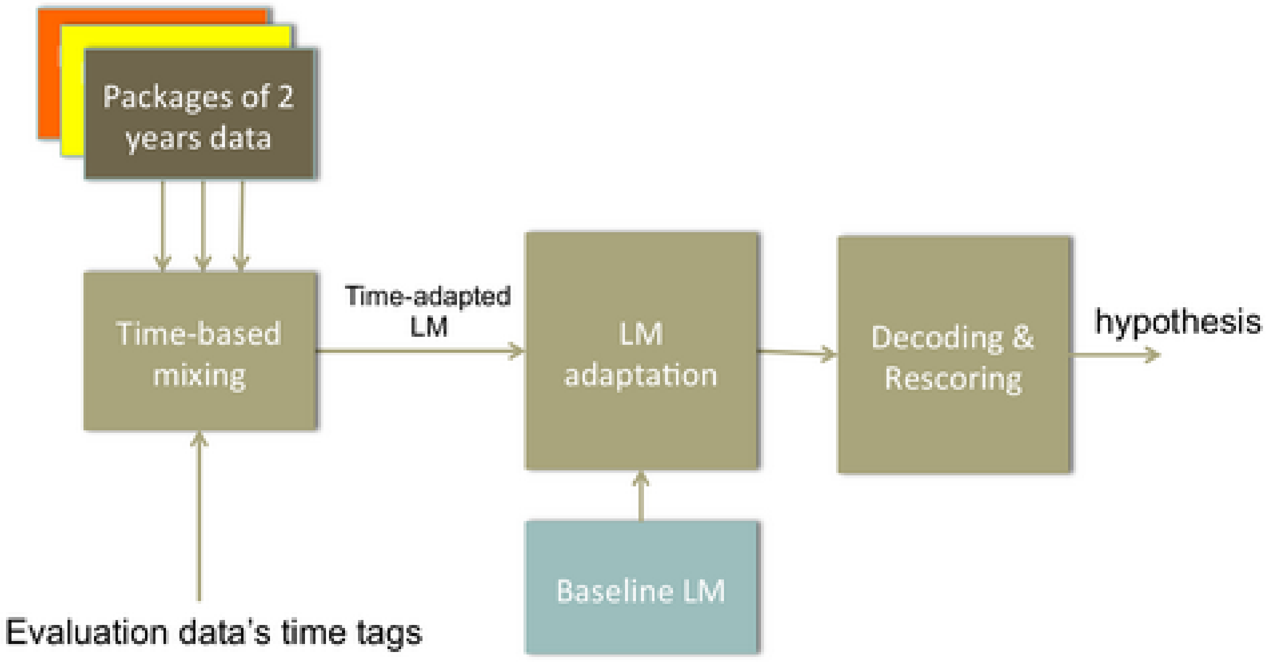
The result of the selection would be a list of words reverse-sorted based on it's estimated count value, for example:

the	56941.91	(in the baseline vocabulary)
to	27383.14	
of	23554.61	
...		
Mccain	57.36	(not in the baseline vocabulary)
Twitter	78.28	

As you can see, some of the resulting words already existed in the previous vocabulary, and some other words are new words that we want to incorporate into the vocabulary. There are two approaches that were taken to use the vocabulary selection:

- ”vocab-regeneration”: Use the same size vocabulary, take top 129135 words from the sorted list. The idea is to incorporate newly occurred words without expanding the vocabulary size.
- ”vocab-addition”: Use baseline vocabulary’s 129135 words and add to it the top 20k from the sorted list which are not yet in the baseline. The final vocabulary would resulted in a 150k words vocabulary. Here the idea is to incorporate new words, but in the same time avoid losing any of the existing words in the baseline vocabulary.

3.2.2 LM adaptation

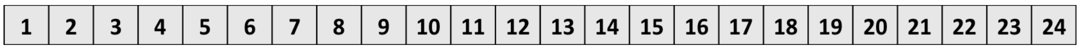


At first the articles and their time tags are grouped into week-sized or month-sized packages measured from each evaluation date. And then the first adaptation is done by mixing those small packages. The resulting LM, called the Time-adapted LM, is then mixed with the baseline LM, to produce the final LM which will then be used for decoding and rescoring which produced hypothesis.

3.2.3 Mixing and packaging the LM

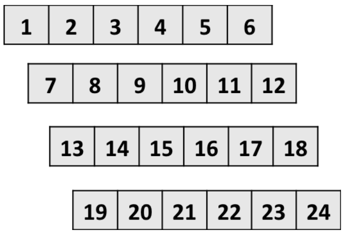
There are several packaging scheme that were tested in this thesis. Each of the following boxes worth 1 month of articles. ”10” means it comes from the tenth months before the evaluation date.

3.2.3.1 Standard 24 months



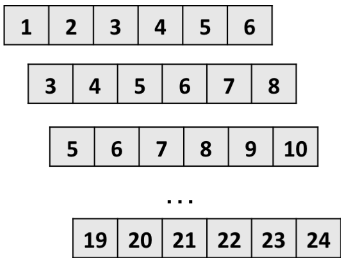
In this scheme all newly available data will be used, which is two years of latest articles before the evaluation date.

3.2.3.2 Mixing isolated 6 months



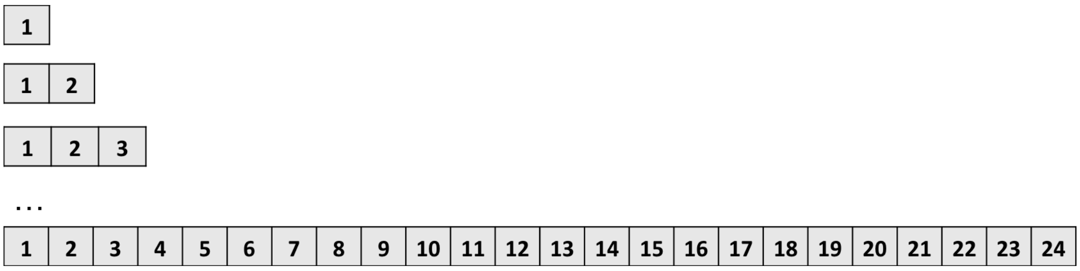
The data is divided into four packages containing 6 months each.

3.2.3.3 Sliding window 6 months



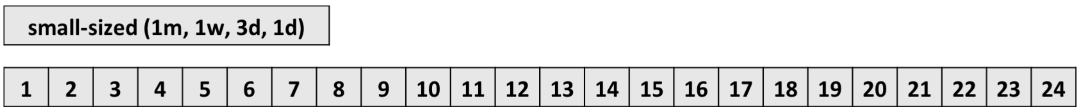
The data is divided into windows of 6 months size, but shifted with 2 month step.

3.2.3.4 Accumulated texts



The data is divided into 24 packages, each containing the the accumulated articles of the last one month, last two months, ..., until the last 24 months.

3.2.3.5 Mix short-term and long-term news



The data is divided into two packages. The first is the short-term data containing articles dated from the last month or shorter period of time which is the closest to the evaluation date. The second is the long-term data containing all 24 months. The difference of this to 3.2.3.4 is that the middle-sized parts were not used in this scheme.

4. Implementation

In this chapter we will discuss various tools, experimental setup, and data formats which are used in this thesis.

4.1 Evaluation Data

The evaluation data consisted of nine shows taken from CNN, BBC, and Channel4 shows, and only breaking news and news shows which contains trending news are selected. CNN and BBC is selected because these are two major news provider which cover major world news stories in great detail. One Channel4 show is selected to measure the performance of adapted LM against a different news channel.

List of shows and the number of sentences for each show:

- 267 channel4_20080709_01_mr
- 407 QRBC_ENG_GB_20100305_145200_NEWSPOD_POD
- 284 QRBC_ENG_GB_20100311_2130_CNN_NEWS
- 273 QRBC_ENG_GB_20100323_2130_CNN_NEWS
- 164 QRBC_ENG_GB_20100325_2130_CNN_NEWS

- 73 QRBC_ENG_GB_20110103_2130_CNN_NEWS
- 87 QRBC_ENG_GB_20110106_2130_CNN_NEWS
- 116 QRBC_ENG_GB_20110110_163000_NEWSPOD_POD
- 154 QRBC_ENG_GB_20110110_2130_CNN_NEWS

The evaluation set contains a total of 1825 sentences and 28830 words

4.2 Toolkits

4.2.1 Janus Recognition Toolkit (Janus)

Janus Recognition Toolkit, also called Janus, is a general purpose speech recognition toolkit developed at the Interactive Systems Labs in Karlsruhe, Germany and Pittsburg, USA. While it's implemented in C, it's encapsulated in Tcl/Tk, so that most of the training and decoding can be done with Tcl/Tk.

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 and re-scoring of lattices in a very efficient way.

4.2.2 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.

4.2.3 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\

4.2.4 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 of the experiments will be presented and evaluated. There are three groups of experiments in this chapter, each of them consisted of series of experiments with different parameters. The first one is experiments regarding vocabulary selection. The latter two contains time-adaptation experiments, using baseline vocabulary and then the selected vocabulary resulted from the first experiment.

5.1 Baseline System

The baseline ASR system used for this thesis is a part of the English ASR developed for the Quaero 2010 English evaluation. The acoustic model is trained based on Mel-frequency Cepstral Coefficients (MFCC) obtained from a fast Fourier Transform [WoMW03]. It provided a feature every 10 ms for training. It applied vocal tract length normalization (VTLN) [ZhWe97], which is done in the linear domain. The MFCC uses 13 cepstral coefficients. Mean and variance of the cepstral coefficients were normalized on a per-utterance basis. 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 [?] was applied, and then Maximum Mutual Information Estimation (MMIE) training to improve the models further.

We take the result of the MFCC 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].

5.1.1 Baseline Language Model

For building the baseline language model used on the existing English ASR system, the corpus English Gigaword 4th Edition, LDC (1994-2008) was used. It contains 159M sentences and 1.8B words. It contains the articles from the following news media:

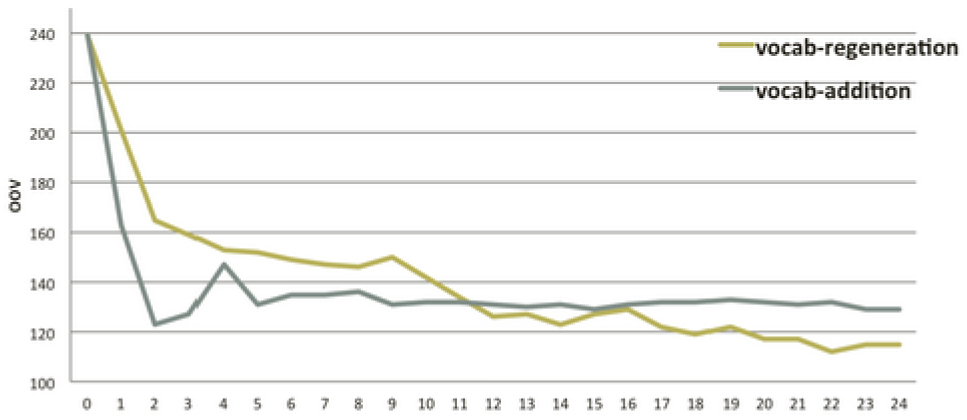
- Agence France-Presse, English Service (afp_eng)
- Associated Press Worldstream, English Service (apw_eng)
- Central News Agency of Taiwan, English Service (cna_eng)
- Los Angeles Times/Washington Post Newswire Service (ltw_eng)
- New York Times Newswire Service (nyt_eng)
- Xinhua News Agency, English Service (xin_eng)

The vocabulary used to build the baseline LM contains 129135 words, which perform quite well and only has 0.82% OOV for all evaluation shows.

5.2 Vocabulary selection

In this section we use vocabulary selection technique described in 2.5.1 and 3.2.1. The selection resulted in a list of vocabulary, which was sorted by a value indicating each of the words' importance. The vocabulary selection was used in two ways:

- ”vocab-regeneration”: Use the same size vocabulary, take top 129135 words from the sorted list
- ”vocab-addition”: Use baseline vocabulary’s 129135 words and add to it the top 20k from the sorted list which are not yet in the baseline. The final vocabulary would resulted in a 150k words vocabulary



Both approaches were tested using the text data that was used to build baseline vocabulary, adding new downloaded data that we got. For each point in x axis, it indicates the numbers of months before the evaluation date, that were used to select articles which were used in vocabulary selection. Zero months means using baseline vocabulary, and the rightmost point in the x axis indicated that all of 24 months of data were used.

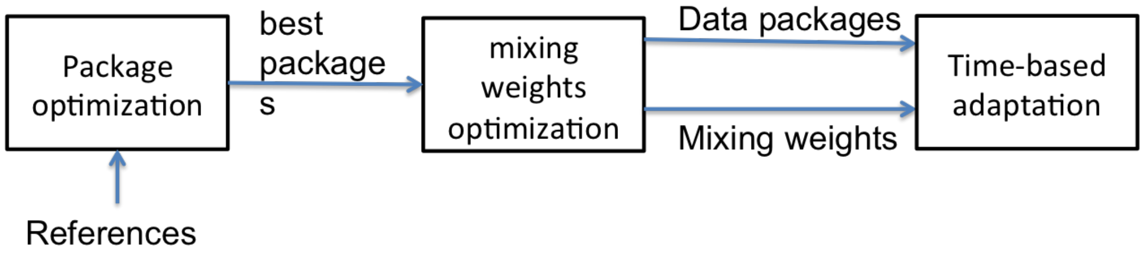
Here we can see that in the long-term the ”vocab-regeneration” approach is the most effective, with OOV reduction from 239 to 115 words, and in the short-term the ”vocab-addition” is the most effective one, with an OOV reduction from 239 to 123 words (using only two months data). From this experiment we can see that using ”vocab-regeneration” or ”vocab-addition” can be decided based on the amount of data that we have. Because we have all 24 months of data, in the time-adaptation experiments with vocab selection, the ”vocab-regeneration” will be used.

5.3 Time-adaptation

To determine the best adaptation method, we have to solve two problems

1. Search for the best data splitting (data packaging)

2. Search for the best mixing weights to mix the packages



To solve the first problem we have to assume that the best mixing weights can be known. In this step an EM algorithm is used, which measure perplexity based on references of the evaluation data. The next step is to find the best mixing weights. In this step we assume the best data packaging already known.

5.3.1 Optimizing the packages

The following experiments were done using the largest show in the evaluation data which contains about 400 sentences: QRBC_ENG_GB_20100305_145200_NEWSPOD_POD

In the following we present the test results of the mixing and packaging schemes that were described more detailed in 3.2.3.

Results:

ID	Packages	Perplexity
STA_24M	1-24	213.877
ISO_6M	1-6; 7-12; 13-18; 19-24	217.887
SLIDE_6M	1-6; 3-8; 5-10; ...; 19-24	216.553
ACCU_24M	1; 1-2; 1-3; ...; 1-24	211.156
SL_1M_24M	1; 1-24	211.487
SL_1W_24M	1w; 1-24	210.48
SL_3D_24M	3d; 1-24	210.448
SL_1D_24M	1d; 1-24	211.151

Here are the list of experiments performed:

1. Standard: take all 24 months (STA_24M)

2. Mixing isolated 6-months texts (ISO_6M)

3. Sliding window size=6m, step=2m (SLIDE_6M)

4. Accumulated texts

   Last 1 month, last 2 months, ..., last 24 months (ACCU_24M)

   Last 1 month, last 2 months, ..., last 12 months (ACCU_12M)

   Last 1 month, last 2 months, ..., last 6 months (ACCU_6M)

5. Mix most recent articles with all data. It is similar to accumulated text experiments, but without the middle part:

   last 1 month + standard 24 months (SL_1M_24M)

   last 1 week + standard 24 months (SL_1W_24M)

   last 3 days + standard 24 months (SL_3D_24M)

   last 1 day + standard 24 months (SL_1D_24M)

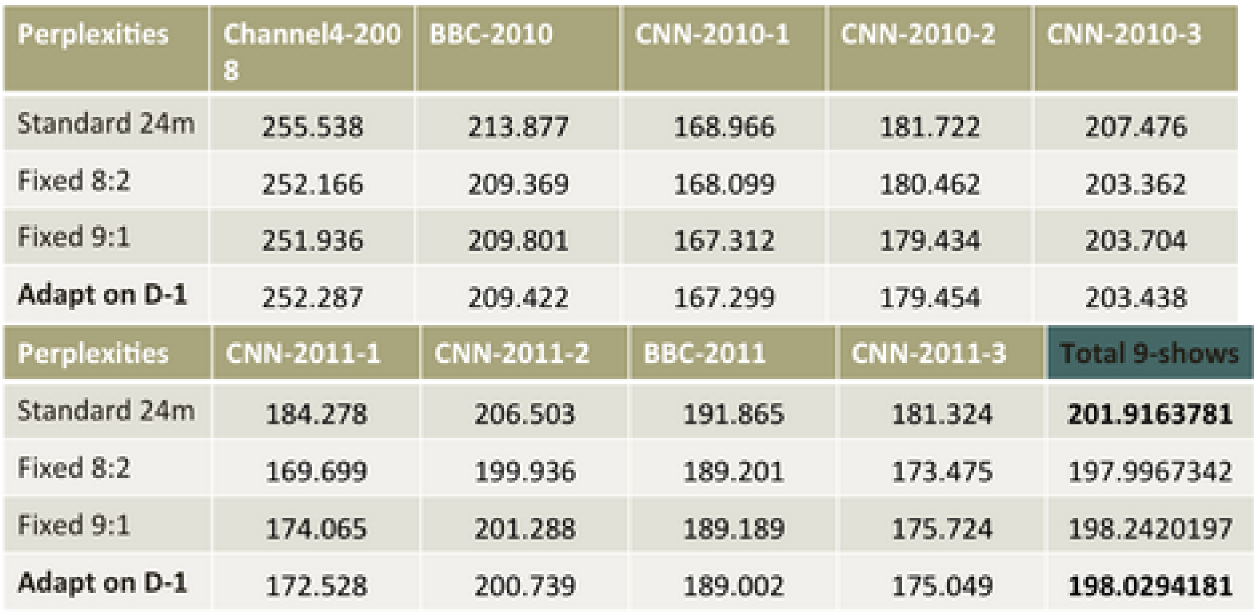
From ACCU_24M experiment, it can be observed that the best mixing weights found by the EM algorithm distributed unevenly, with most of the weights clumped on the left and right, which means on all (long-term) data, and last short-term data. When the experiment is repeated with different sizes of long-term data, the same phenomenon occurred.

ID	mixing weights
ACCU_24M	(0.157 0.008 … 0.168 0.459)
ACCU_12M	(0.099 0.058 … 0.079 0.577)
ACCU_6M	(0.098 0.047 0.032 0.042 0.127 0.652)

From the experiment we can observe that the best result was achieved by mixing between small-sized most recent articles, mixed with all data that we have (two years of data before the evaluation date).

5.3.2 Optimizing the mixing weights

In this section a packaging of the data into two parts has been decided. One part contains the most recent three days LM, and the second part is an LM built using all two years data before the evaluation date. Using this mixing approach, the best mixing weights can be found.



Perplexities	Channel4-2008	BBC-2010	CNN-2010-1	CNN-2010-2	CNN-2010-3
Standard 24m	255.538	213.877	168.966	181.722	207.476
Fixed 8:2	252.166	209.369	168.099	180.462	203.362
Fixed 9:1	251.936	209.801	167.312	179.434	203.704
Adapt on D-1	252.287	209.422	167.299	179.454	203.438
Perplexities	**CNN-2011-1**	**CNN-2011-2**	**BBC-2011**	**CNN-2011-3**	**Total 9-shows**
Standard 24m	184.278	206.503	191.865	181.324	**201.9163781**
Fixed 8:2	169.699	199.936	189.201	173.475	197.9967342
Fixed 9:1	174.065	201.288	189.189	175.724	198.2420197
Adapt on D-1	172.528	200.739	189.002	175.049	**198.0294181**

The above table listed result from each of the nine evaluation shows and one using a total (weighted average) of all shows. There are four types of measurement for each column: the standard way of using all 24 months of data before evaluation date; using time-adapted approach with 8:2 mixing ratio; using the same but with 9:1 ratio; and using the ”Adapt on D-1” approach described in 2.4.2. From the averaged perplexity results, the fixed mixing with 8:2 ratio returns the slightly best result when averaged for all shows, but overall the ”Adapt on D-1” approach is quite robust to approximate best mixing weights for most of the shows. It returns 1.9% perplexity reduction compared to the one using all 24 months data.

5.3.3 WER experiments

The baseline system using the baseline Gigaword LM resulted in WER 25.67%

- Using LM adaptation and baseline vocabulary, it resulted in:
  WER 25.16% (0.51% improvement)

- Using LM adaptation and “vocab-regeneration” adaptation, it resulted in:
  WER 25.09% (0.58% improvement)

The WER results were measured for all nine shows, decoding each of the shows separately, each using separately adapted LM.

6. Summary

From the experiments in this work, several conclusions can be drawn. The best OOV reduction was achieved using Venkataraman and Wang's vocabulary selection technique ("vocab-regeneration" approach): a reduction from 0.82% to 0.39% OOV. If the data available is much smaller however, better result can be achieved by using "vocab-addition" approach with small amount of most recent data (2-3 months). This vocabulary selection, combined with the time adaptation method using 3-days mixed with all 24-months data has improved the system by 0.58% absolute WER improvement.

6.1 Further works

In this work only few simple adaptation technique has been tested. Different adaptation and mixing techniques should be tested further. Also CNN and BBC shows were merged and used as a training data together, while each of them might have different trending news (CNN is more US-oriented and BBC is UK-oriented). Different news can become trending with different segment of the society. Further information such as geographical, gender, age, interest, and other information nowadays made possible by the social media, can improve the system even better by restricting the domain further.

Bibliography

[CLGA04] L. Chen, L. Lamel, J.-L. Gauvain und G. Adda. Dynamic language modeling for broadcast news. In *INTERSPEECH*. ISCA, 2004.

[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.

[KOIA98] A. Kobayashi, K. Onoe, T. Imai und A. Ando. Time dependent language model for broadcast news transcription and its post-correction. In *ICSLP*. ISCA, 1998.

[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.

[VeWa03] A. Venkataraman und W. Wang. Techniques for effective vocabulary selection. *CoRR* Band cs.CL/0306022, 2003.

[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.