UNIVERSITY OF LORRAINE
SAARLAND UNIVERSITY

MASTER'S THESIS

Language Model Contextualization for Automatic Speech Recognition by Dynamic Adjustment

Author: Anna CURREY

Supervisors: Dominique FOHR
Irina ILLINA
Dietrich KLAKOW
Jury: Maxime AMBLARD
Philippe DE GROOTE

A thesis submitted in fulfillment of the requirements for the degree of

Master of Science in Language and Communication Technologies

Based on work done during an internship with the

Multispeech Team
Loria, Inria, and University of Lorraine
February 1, 2016 – June 15, 2016

July 1, 2016

Declaration of Authorship

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Signed:

Date:

UNIVERSITY OF LORRAINE AND SAARLAND UNIVERSITY

Abstract

Multispeech Team
Loria, Inria, and University of Lorraine

Master of Science in Language and Communication Technologies

Language Model Contextualization for Automatic Speech Recognition by Dynamic Adjustment

by Anna CURREY

Out-of-vocabulary (OOV) words can pose a particular problem for automatic speech recognition (ASR) of broadcast news. The language models (LMs) of ASR systems are typically trained on static corpora, whereas new words (particularly new proper names) are continually introduced in the media. Additionally, such OOVs are often content-rich proper nouns that are vital to understanding the topic. In this work, we explore methods for dynamically adding OOVs to language models by directly estimating their language model parameters. We concentrate on adaptation of the bigram language model used in the first pass of our ASR system.

We propose two strategies for estimation of n-gram parameters. The first relies on finding in-vocabulary (IV) words similar to the OOVs; OOV behavior is modeled after the behavior of these similar IVs. We use word embeddings to define similarity and examine various word vector training architectures, finding skip-gram with a context size of two words to work best for our application.

Our second strategy leverages a small contemporary corpus to estimate OOV unigram probabilities and to find bigrams containing the OOVs. We find that it is best to limit the number of bigrams added to the language model; this allows us to increase the coverage of the model without creating a significant amount of noise.

We use two experimental setups to evaluate our proposed methods. In experiment 1, we create only one adapted LM for each algorithm; these LMs are then incorporated into our existing ASR system and recognition error rates (word and proper noun) are calculated. In experiment 2, we create a separate language model for each article using each of the algorithms and find the average perplexity over all of the articles. In both experiments, our adapted models improve over the baseline in perplexity; improvements are greatest in experiment 2. Only the corpus-based adaptation method improves significantly over the baseline in recognition error rate.

Keywords: ASR, domain adaptation, language modeling, OOV, word embeddings

Acknowledgements

First and foremost, I would like to thank my supervisors, Irina Illina, Dominique Fohr, and Dietrich Klakow, for their invaluable help and guidance on this project. I would also like to gratefully acknowledge the help of the Multispeech team at Loria, particularly Imran Sheikh.

On a personal level, I would like to thank my family and friends for their love and support, without which this thesis would not have been possible.

Contents

List of Tables

List of Abbreviations

APP	**A**djusted **P**er**p**lexity
ASR	**A**utomatic **S**peech **R**ecognition
CBOW	**C**ontinuous **B**ag **o**f **W**ords
CRF	**C**onditional **R**andom **F**ields
IV	**I**n **V**ocabulary
KL divergence	**K**ullback-**L**eibler divergence
LM	**L**anguage **M**odel
NLP	**N**atural **L**anguage **P**rocessing
NN	**N**eural **N**etwork
OOV	**O**ut **o**f **V**ocabulary
PNER	**P**roper **N**oun **E**rror **R**ate
PP	**P**er**p**lexity
UNK	**Unk**nown
WER	**W**ord **E**rror **R**ate

Chapter 1

Introduction

1.1 Motivation

Automatic speech recognition (ASR) is a subfield of natural language processing (NLP) with many practical applications. In particular, recognition of broadcast news speech can be important for information retrieval and information extraction. The automatic transcription of broadcast news can also allow people who are unable to listen to a broadcast to gain access to the information it contains. Therefore, improving recognition of news speech is an interesting theoretical and practical problem.

Automatic speech recognition systems are often trained on large but static text corpora and with a fixed vocabulary. For a system whose goal is to recognize newly produced speech, particularly speech about current events, this can pose a problem, since new words are continually introduced based on the events that occur. A particular issue is proper nouns: names of newly important people or locations may not be in the vocabulary of the system, but recognizing them can be paramount to understanding the topic. Note that it is not possible to simply find a large enough corpus or lexicon to cover all of the important words, as novel words and names will always be introduced into a language. In addition, treating these words as 'unknown' seems sub-optimal, since they may be essential to understanding the content of the speech.

Therefore, a competent ASR system dealing with current events will have to be able to accommodate adding new words to its vocabulary on the fly. Doing so will allow it to recognize speech about different topics and named entities, including those that were not present in the training data. This requires adjustments to the language model (LM), which tells the system how likely a hypothesized sequence of words is, and to the pronunciation lexicon, which tells the system how a word is pronounced; updating the language model is the focus of this thesis. Note that the acoustic model of the ASR system does not need to be adjusted.

The goal of this master's thesis is to explore different methods for dynamically updating the language model of an ASR system to accommodate new words. The approaches that are investigated were inspired by a number of sub-fields of NLP, including statistical language modeling, continuous representations of words, and domain adaptation. We propose multiple possible solutions for incorporating new words into a language model for ASR dynamically.

1.2 About the project

This thesis was done within the framework of the ANR project ContNomina (Contextualization for Proper Noun Recognition in Diachronic Audio Documents). The overall goal of this project is to improve the recognition of proper nouns in broadcast news audio. For each news article, this is done as follows. First, a list of out-of-vocabulary (OOV) words relevant to a given article is generated based on the initial hypothesis of the ASR system. These words are then added to the system's lexicon and language models.

The role of this thesis within the larger project is to work on adding topic-relevant words to the language models. In particular, we concentrate on the bigram language model, although extensions for other language model types are discussed as well. To learn more about the methodology proposed for finding relevant OOVs, see Sheikh et al. (2015a), Sheikh et al. (2015b), and Sheikh et al. (2016a).

1.3 Research question

As explained above, it is important for the language models of an ASR system to be able to model content-relevant proper nouns, even if those words were not part of the original vocabulary. Therefore, this thesis explores ways of dynamically updating language models by adding words to their vocabularies. This task has many practical applications; for example, a computer-aided captioning system may want to allow users to add new words, either beforehand or when correcting the automatic output. In addition, an audio indexing system may benefit from having a general language model, from which a new model is created with relevant OOV proper nouns each time an article is added to the archive.

In particular, this thesis explores ways of updating the bigram language model used in the first pass of the ASR system. The methods developed in this thesis start from a list of topic-relevant words and estimate probabilities of n-grams containing those words. In order to make this task as realistic as possible, the word lists, pronunciations, and word similarity measures are automatically generated, and more topic-relevant words are added than needed (privileging recall over accuracy or precision).

A few requirements for this task have been taken into account when designing the methods. First of all, a separate new language model is usually created for each article, since each article has a different list of topic-relevant OOVs. In addition, the language models must be updated dynamically; it is not possible to retrain the models or train new models, since the new OOVs should be recognized in real-time. Similarly, very little data (or none at all) is available to add words to the language models. As a result, in this thesis we concentrate on methods of direct estimation using word similarity and a small contemporary contexts corpus.

1.4 Contributions

We make a number of contributions to the task of dynamically adding OOV words to a language model without retraining the model. We recommend methods for estimating language model parameters using only word similarity, without the help of other outside resources. We also describe how to leverage a small contemporary corpus to update the parameters of a language model trained on data from a different time period. The approaches proposed for both of these frameworks result in an improvement over the baseline in perplexity; in addition, the approaches using a small contemporary corpus yield a significant improvement over the baseline in word error rate.

In addition to our implemented methods, we also suggest and describe various other strategies that could be used to enhance LM estimation in this task. These include different ways of renormalizing the language models and a novel word similarity measure that may be more appropriate for this task. Finally, we specify how to extend the approaches that have been proposed for bigram LMs to higher-order n-gram models.

1.5 Outline of the thesis

This thesis is organized as follows. In chapter 2, we give the necessary background information to understand the approaches that we have implemented and describe other strategies that have been used to solve the problem of dynamically updating language models. Then, chapter 3 describes the methods that we have explored in the experiments, as well as some ideas for additional approaches that could be used. The setup of the experiments, including the data and the evaluation measures used, is described in chapter 4; this is followed in chapter 5 by an explanation of the experiments and their results. Finally, chapter 6 discusses the results and suggests some directions for future work.

Chapter 2

Review of Literature

This chapter contains an overview of topics that are relevant to this master's thesis. The goals of the chapter are twofold. First, we want to provide the basic background information necessary to understand the methodology and discussion of the present work. Second, we intend to give an idea of current and interesting research being done in related areas, as well as techniques that have already been proposed to solve the problem at hand.

2.1 Background

2.1.1 Automatic speech recognition

The goal of automatic speech recognition is to find the sequence of natural language units (e.g. words) that is the most likely match of a given acoustic signal. This can be formalized with the following equation (Shannon, 2001):

$$W^* = \operatorname*{argmax}_W P(W|X) \tag{2.1}$$

where W is a sequence of words, X is the acoustic signal, and W^* is the most likely sequence of words (Bahl, Jelinek, and Mercer, 1983). Applying Bayes's theorem to equation (2.1) and reducing gives:

$$W^* = \operatorname*{argmax}_W P(X|W)P(W) \tag{2.2}$$

$P(X|W)$ in equation (2.2) can be conceptualized as the likelihood of producing acoustic signal X to convey sequence W; in ASR, this is referred to as the acoustic model. $P(W)$ is the language model; it models the likelihood of a sequence W occurring in the language. (Language model estimation strategies for ASR are the main focus of this thesis and will be discussed in more depth in section 2.1.2.)

In practice, language models in current ASR systems are often used in more than one place (Sundermeyer, Ney, and Schlüter, 2015). First, a language model (in our system, a bigram LM) is used to generate a lattice of hypotheses of word sequences. Afterwards, a more advanced language model (in our system, a 4-gram LM and a recurrent neural network LM) is used to rescore the lattice and select the best hypothesis.

2.1.2 Statistical language modeling

Statistical language modeling is the process of estimating the probability distribution over sequences of words or other linguistic units in a language. Language models have many practical applications in natural language processing. Besides being integral to automatic speech recognition, language models are beneficial for statistical machine translation (Brown et al., 1990), information retrieval (Ponte and Croft, 1998), optical character recognition (Tong and Evans, 1996), spelling correction (Kukich, 1992), and other tasks. In this section, we will give a background on statistical language modeling, concentrating on n-gram models (also known as count-based models), since those are the models that are adapted in this thesis. Only those language modeling techniques that are most relevant to the task at hand will be discussed here; Rosenfeld (2000) gives a more complete review.

Most traditional count-based language models estimate the probability of a sequence using the history of each word (Rosenfeld, 2000):

$$P(w_1, ..., w_n) = \prod_{i=1}^{n} P(w_i|h_i) \tag{2.3}$$

where w_i is the i^{th} word in a sequence and h_i is the history of w_i. One of the most popular techniques in language modeling is n-gram language models. These models rely on the Markov assumption that the occurrence of a word depends only on the previous $n-1$ words, so that we can model $P(w_i|h_i)$ from equation (2.3) as:

$$P(w_i|h_i) \approx P(w_i|w_{i-n+1}, ..., w_{i-1}) \tag{2.4}$$

These models have been quite successful and were difficult to improve upon until the introduction of neural language models (Bengio et al., 2003). Despite the improvements made by neural language models, n-gram models are still commonly used due to their size and simplicity, especially in the first pass of ASR (Sundermeyer, Ney, and Schlüter, 2015). In particular, dynamically adapting a bigram language model is the focus of this thesis, since that is the model used to generate the lattices in our ASR system.

Count-based language models are typically calculated using a modified maximum likelihood estimate on large corpora. Since it is impossible to have a corpus large enough that it contains examples of every possible sequence, using a pure maximum likelihood approach would result in zero probabilities given to possible sequences. As a result, the maximum likelihood probabilities have to be smoothed. Various approaches to smoothing n-gram language models have been proposed. Here, we will only discuss modified Kneser-Ney smoothing (Chen and Goodman, 1999; Ney, Essen, and Kneser, 1995), as it is one of the most successful smoothing methods (Chen and Goodman, 1999) and the one used in our system. For a comprehensive survey of language model smoothing techniques, see Chen and Goodman (1999). The insight that modified Kneser-Ney smoothing captures is that words with more diverse histories in the training data are more likely to show up in novel contexts in the test data. To model this, the unigram probability of a word is estimated as:

$$P(w_i) = \frac{|w_{i-1} : N(w_{i-1}w_i) > 0|}{\sum_{w_j} |w_{j-1} : N(w_{j-1}w_j) > 0|} \tag{2.5}$$

where $N(w_{i-1}w_i)$ is the frequency of the sequence $w_{i-1}w_i$. This smoothing method then interpolates higher-order and lower-order n-grams to estimate the probability of a sequence, using absolute discounting based on the count of the sequence. The resulting estimate of an n-gram probability with modified Kneser-Ney smoothing is thus as follows:

$$\begin{aligned} P(w_i|w_{i-n+1}, ..., w_{i-1}) = {} & \frac{\max(N(w_{i-n+1}...w_i) - \delta, 0)}{\sum_{w_j} N(w_{j-n+1}...w_j)} \\ & + \frac{\delta}{\sum_{w_j} N(w_{j-n+1}...w_j)} |w_j : N(w_{i-n+1}...w_{i-1}w_j) > 0| P(w_i|w_{i-n+2}, ..., w_{i-1}) \end{aligned} \tag{2.6}$$

Here, δ is a discounting factor that is estimated based on development data; it generally takes three values: one for sequences with a count of one, one for sequences with a count of two, and one for all other observed sequences (Chen and Goodman, 1999).

Language models are often restricted to a given vocabulary, because the model will not be able to estimate rare words well given the training data or because the model is part of a larger ASR system, for example. When this is the case, any word in the training data that is not in the vocabulary is replaced with a special token, <unk> (for 'unknown'). The language model is then estimated as described above, but all <unk>s are treated as one word and the appropriate probabilities are calculated for that word. When the model is queried, individual out-of-vocabulary words cannot be predicted, but the presence of a <unk> can.

In practice, researchers have found that it is often useful to combine multiple language modeling techniques (Goodman, 2001). Many methods encode orthogonal information, so combining them into one model allows that model to take advantage of more information about the language. Interpolation of language models can be done linearly or log-linearly (Klakow, 1998), on the model level or on the count level (Bellegarda, 2004), and based on histories or generally (Kuhn and De Mori, 1990). Maximum-entropy models can also be used to combine language models with information from other sources (Rosenfeld, 1996).

2.1.3 Language model evaluation

Perplexity (PP) is one of the most widely-used metrics for evaluating language models due to the fact that it can be calculated easily and independently of a speech recognition system (Chen, Beeferman, and Rosenfeld, 1998). Given a language model $P(w_i|h_i)$ and a test corpus, the perplexity of the LM on the test corpus can be calculated as follows (Bahl, Jelinek, and Mercer, 1983; Klakow and Peters, 2002):

$$PP = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|h_i)\right) \tag{2.7}$$

where N is the number of words in the test corpus, w_i is the i^{th} word, and h_i is the history of w_i. Clearly, a lower perplexity implies a better probabilistic model of the test data, so the goal in language modeling is often to minimize perplexity. It should be noted that perplexity can only be calculated for language models that represent probability distributions. In addition, the perplexities of two language models with different vocabularies cannot be compared (Chen, Beeferman, and Rosenfeld, 1998); this is because language models with larger vocabularies may be unfairly penalized for trying to model more words (Ueberla, 1994). To deal with this issue, Ueberla (1994) introduced an adjusted perplexity (APP) measure, formulated by Naptali, Tsuchiya, and Nakagawa (2012) as:

$$APP = (P(W)m^{-o})^{-\frac{1}{N}} \tag{2.8}$$

where $P(W)$ is the probability of the test corpus under the language model, m is the number of OOV types in the training data, o is the number of OOVs in the test corpus, and N is the number of words in the test corpus.

For evaluating entire ASR systems, word error rate (WER) is often used. Word error rate is based on a comparison of the system's transcription to the gold standard transcription and is calculated as follows (Johnson et al., 1999):

$$WER = \frac{S+I+D}{W} \tag{2.9}$$

where S, I, and D are the number of substitution, insertion, and deletion errors, respectively, and W is the number of words in the gold standard.

Given that perplexity is often used both as a quality measure and as an optimization target for language models, a natural question to ask is how well it actually approximates word error rate. Answering this question is not trivial, and results in the literature have been inconclusive. Sundermeyer, Ney, and Schlüter (2015) found a correlation between the two measures for many different types of language models, while Klakow and Peters (2002) found that the two measures were related by a power law. It is empirically true, though, that two language models with identical perplexities on a test set may not have identical word error rates on that test set (Klakow and Peters, 2002), and other authors have reported that perplexity and word error rate do not correlate well (Chen, Beeferman, and Rosenfeld, 1998; Clarkson and Robinson, 1998; Iyer, Ostendorf, and Meteer, 1997). As a result, a number of alternatives have been proposed that take into account additional characteristics of the language model (Chen, Beeferman, and Rosenfeld, 1998), use decision trees to compare two ASR systems (Iyer, Ostendorf, and Meteer, 1997), apply word error rate calculations to artificially generated lattices (Chen, Beeferman, and Rosenfeld, 1998), or exploit the difference in probability between the word predicted by the language model and the actual word in the test data (Ito, Kohda, and Ostendorf, 1999).

Although word error rate as a measure of the performance of an ASR system makes intuitive sense, it is not without flaws, and some alternatives have been suggested. For example, Wang, Acero, and Chelba (2003) showed that although their system performed worse in word error rate, it had a higher understanding accuracy. Sandness and Hetherington (2000) offered keyword error rate as a possible alternative to word error rate and found that lowering keyword error rate also lowered understanding error rate. However, Park et al. (2008) did not find a significant difference between keyword error rate and WER. Similarly, Nanjo and Kawahara (2005) proposed a weighted keyword error rate that used automatically learned weights based on word importance. However, despite the issues and proposed improvements, perplexity and word error rate remain the most common standards for evaluating language model and speech recognition system performance.

2.1.4 Continuous representations of words

Many techniques for dynamically adding OOVs to language models rely on measures of similarity between the OOVs to be added and the in-vocabulary (IV) words. Finding similar IV words can help to model the expected behavior of a given OOV. This section contains information on how word similarity can be measured, concentrating on continuous word representations.

In many traditional NLP approaches, including the language models described in section 2.1.2, words are treated as atomic units. This simplifying assumption hinders the ability of the models to generalize over related words, which means that rare words, in particular, are not modeled as well as they could be. One response to this issue has been to introduce multidimensional vector representations of words, in which the dimensions model different types of similarity over words; here, we discuss various methods for estimating these vectors. Once continuous word representations are found, finding similarity between words is simply a matter of measuring the distance between the word vectors.

Most attempts at representing words as vectors use the distributional hypothesis of Harris (1954) that two words that tend to appear in similar contexts will have similar meanings; this was empirically corroborated by Rubenstein and Goodenough (1965). Until relatively recently, most word vectors were based on counts of words, contexts, or other features; here, we will refer to these as count methods, following Baroni, Dinu, and Kruszewski (2014). Turney and Pantel (2010) divide such models into three classes based on the matrices used: term-document, word-context, and pair-pattern. Term-document matrices consist of rows corresponding to terms and columns corresponding to documents. Word-context matrices are similar, but use specified contexts as columns instead of documents. In pair-pattern matrices, rows contain word pairs and columns contain patterns in which the pairs can occur. The entries can be simple frequencies or they can be weighted in various ways; there are also many other parameters, such as smoothing methods and linguistic processing of the text. The best strategy for creating word vectors will probably differ depending on the application and on the data available, although Bullinaria and Levy (2007) found that using word-context matrices with positive pointwise mutual information worked particularly well on their semantic and syntactic tasks.

A relatively recent trend in this area is the use of predictive models, also called word embeddings, to train word vectors. In fact, neural network language models implicitly train and use word embeddings; the word representations produced as a side effect of neural network (NN) LMs can subsequently be used in other applications. For example, Collobert et al. (2011) trained word vectors using a neural network with one hidden layer; these word vectors were then used in various tasks, including the training of a neural network language model. Similarly, Mikolov, Yih, and Zweig (2013) examined the vectors resulting from a recurrent neural network LM and found that they encoded both semantic and syntactic properties of the words. However, word embeddings do not necessarily have to be trained using neural networks. One of the most popular toolkits for creating word embeddings is word2vec (Mikolov et al., 2013b), which simplifies the neural network architecture by removing the non-linear hidden layer. word2vec contains two separate models for estimating the vectors: the continuous bag of words (CBOW) model takes as input the left and right contexts and tries to predict the current word, while the skip-gram model attempts to predict the context words given the current word. The latter was later improved with negative sampling (Mikolov et al., 2013a), which is an alternative to the softmax function that adds negative examples for each positive example in the data. Another toolkit for learning word embeddings is GloVe (Pennington, Socher, and Manning, 2014), which uses statistics about the entire corpus in training (rather than just about a local context). Other predictive strategies for creating word representations use canonical correlation analysis (Dhillon, Foster, and Ungar, 2011) or noise-contrastive estimation (Mnih and Kavukcuoglu, 2013).

Initial experiments on predictive word embeddings showed that they were better models of word similarity than traditional count-based methods. Indeed, a systematic comparison of count-based methods to the CBOW model by Baroni, Dinu, and Kruszewski (2014) found that the latter outperformed the former in most tasks. Analysis by Levy and Goldberg (2014b), however, showed that the skip-gram with negative sampling of Mikolov et al. (2013a) is an implicit factorization of a shifted word-context pointwise mutual information matrix. It was later shown that the advantages of predictive models were due to the selection of hyperparameters (Levy, Goldberg, and Dagan, 2015). However, the word2vec model was more computationally efficient than the other models examined (Levy, Goldberg, and Dagan, 2015). It is also important to mention that these models largely encode only taxonomic semantic information (Rubinstein et al., 2015), so they may not be optimal for all NLP tasks.

Following the success of these predictive methods for estimating word vectors, many extensions have been proposed. Levy and Goldberg (2014a) described a modification of the skip-gram with negative sampling model of Mikolov et al. (2013a) that used syntactic dependency-based contexts instead of a continuous context of a fixed length. As expected, the resulting word vectors encoded more syntactic information than in the original model. A similar technique was used by Bansal, Gimpel, and Livescu (2014), who also argued that using a smaller context window caused the vectors to be more syntactic than semantic in nature. Methods for augmenting the word vectors with morphological information have also been explored, notably by Botha and Blunsom (2014) and Luong, Socher, and Manning (2013); these techniques showed improved performances in statistical machine translation and word similarity detection, respectively. It should be noted, however, that such models often rely on separate syntactic or morphological analyzers. Finally, Ling et al. (2015) modified the word2vec toolkit to create the wang2vec tool, which augmented the existing skip-gram and CBOW models to include word position information. Although these models need to be trained on a lot of data (on the order of 100 million words) to prevent data sparsity issues, the authors argued that the resulting vectors were more syntactic than the original word2vec vectors; indeed, they performed better on part-of-speech tagging and dependency parsing tasks.

2.2 State of the art

Language models are often trained on huge amounts of general data, which can make them sub-optimal when applied to a specific domain, since different domains can have different vocabularies, styles, and structures. Central to this thesis is the adaptation of language models to a specific time period, which offers similar challenges. In this section, we discuss some strategies that have been proposed to deal with the problem of language model adaptation, concentrating on methods for adding domain-specific vocabulary to a model without having to retrain it. Many other approaches to language model adaptation exist; for a comprehensive review of adaptation techniques applied to count-based language models, see Bellegarda (2004).

2.2.1 Direct estimation for LM adaptation

An important strategy for updating existing n-gram language models with domain-relevant OOV words consists of adding n-grams containing the relevant OOVs directly to the language model. Many such approaches rely on a notion of similarity between OOV and IV words for adapting the LM. For example, Lecorvé, Gravier, and Sébillot (2011) defined equivalence relations for words and for n-grams based on part of speech and semantic relatedness. For each OOV word, they added the novel n-grams containing that word that were equivalent to existing n-grams in the model. Orosanu and Jouvet (2015) also directly defined n-grams for relevant OOV words, this time using a small amount of sentences to find distributionally similar in-vocabulary words based on Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951). The new n-grams were added to the model by replacing similar in-vocabulary words with new OOV words and keeping the same probabilities; the probabilities were renormalized once the new words were added. It should be noted that all such available n-grams were added to the model. Finally, Qin (2013) suggested generating n-grams to add to a language model by taking an OOV word's context and replacing the surrounding in-vocabulary words with other similar in-vocabulary words.

2.2.2 Class-based adaptation

Another common strategy for dynamic adaptation of count-based language models is derived from class-based language modeling. The basic idea behind class-based adaptation is to classify an OOV word and then assign LM probabilities based on this classification to allow domain-relevant OOVs to be added to an existing LM. This strategy can be seen as a generalization of a standard language model using <unk> tags (Allauzen and Gauvain, 2005); instead of assigning probability mass to a single <unk> word, the model includes multiple <unk>s corresponding to the different word classes. Hence, n-grams are directly estimated to create the adapted models, although the original language models have been expressly trained in such a way to allow this. This strategy was introduced by Allauzen and Gauvain (2005), who trained a standard language model with twenty additional OOV word classes. The probability of a word given its history was then generated as in a class-based n-gram model, with each in-vocabulary word belonging to its own class. The OOV word classes and emission probabilities were estimated using the adaptation data. Extensions to the class-based adaptation model include using linguistic information to better inform classes (Pražák, Ircing, and Müller, 2007) and using word vectors to cluster OOV words into classes (Naptali, Tsuchiya, and Nakagawa, 2012). In addition, Martins, Teixeira, and Neto (2008) experimented with updating a language model without any adaptation data. They achieved this by classifying unseen words using morpho-syntactic information. They then gave each OOV word the probability of the most probable word in its class; all probabilities were then renormalized and smoothed.

2.2.3 Interpolation

Language model interpolation is another effective domain adaptation strategy; a general language model can be interpolated with a language model trained on in-domain data. This interpolation can be done at different levels, including n-gram count, model, and topic levels (Ma et al., 2014). For example, in Yamazaki et al. (2007), n-gram counts from presentation slides were interpolated with the counts from the general language model to improve OOV recognition in an ASR application. Liu, Gales, and Woodland (2013) performed a detailed study on the options for interpolation, including methods for combining models and for tuning the weights. They suggested adapting the interpolation weights based on the history, and they compared methods for tuning interpolation weights that minimize perplexity to those that minimize error rate as modeled by the minimum Bayes-risk criterion, finding that the latter performed better. In addition, Ma et al. (2014) found that linear interpolation at the model level based on n-gram clustering performed better than linear regression and direct estimation.

2.2.4 Summary

This thesis is inspired by some of the ideas mentioned in this section, particularly using word similarity to estimate n-gram probability and using a small amount of data to find n-grams. However, most previous research has either added only a few new words per language model or added only words to the language model that were known to be in the test data. It is rather easy to improve a model in this case, since an artificially high probability can be given to such words based on the confidence that they will occur. The work done here, though, is part of a larger system, with many automatic and imperfect components. Since language models give probabilities, and total probability mass is fixed, it is important to strike a balance between giving a high probability to the relevant OOV words that occur and not taking too much probability from existing IVs to give to OOVs that are not in the test data. The methods proposed in this thesis attempt to effectively assign appropriate probabilities given this experimental framework.

Chapter 3

Proposed Approaches

3.1 Overview

In this chapter, we give a detailed description of our proposed approaches for directly adding OOV proper nouns to a language model. Since these OOVs become IVs for the language model that they are added to, we will refer to them as *new IVs*. OOVs that are not added to the language model will still be referred to as *OOVs*, while words in the vocabulary of the original language model will continue to be referred to as *IVs*. The overall process for adding new IVs to our system for a given news article is as follows:

1. The recognition system is run to create a hypothesis for the article
2. Based on that hypothesis, a list of new IVs that are likely to be in the article is generated
3. A grapheme-to-phoneme conversion is applied to assign pronunciations to the new IVs
4. The new IVs are added to the lexicon
5. The new IVs are added to the language model and the parameters of the language model are re-estimated

In this thesis, we address adding the new IVs to the language model (step 5). In principle, the approaches described in this chapter could be applied in any situation where new IVs are added to a language model, regardless of the method for generating the list of hypothesized new IVs or the new IV pronunciations. For more information about the new IV retrieval methods, see section 4.2.

Thus, for the purposes of this chapter, we will simply assume that we have the following input:

- A baseline bigram language model
- A list of new IVs to be added to the language model for each article
- A word similarity table (see section 3.2)
- For some approaches (see sections 3.3.2 and 3.3.3), a corpus of textual data contemporary to the article to be recognized; we will refer to this as the *contemporary corpus*

From this input, our goal is to apply our estimation approaches to create a new language model that contains the vocabulary of the original language model, as well as the new IVs. Ultimately, our proposed approaches would be used to create a separate language model for each news article.

Our approaches for adding new IVs to a language model take advantage of similarity between new IVs and existing IV words (referred to here as *word similarity*); in section 3.2, we describe the different word similarity measures that we study in this work. Section 3.3 contains a specification of our proposed approaches for language model parameter estimation, and section 3.4 gives information on how the adapted language models are renormalized. Finally, we conclude this chapter by giving information about additional possible approaches that have not yet been implemented.

3.2 Word similarity measures

All of our proposed approaches leverage word similarity to estimate language model parameters at least in some way; we find IV words that are similar to a given new IV and use the behavior of those IV words to aid in the modeling of the behavior of the new IV. It is important to keep in mind that many different word similarity measures could be used in our approaches, including measures based on the various continuous word representations described in section 2.1.4; we propose an additional word similarity measure in section 3.5.3. In our experiments, we study word similarity measures based on word embeddings; this section describes the measures used and their variations.

For finding IVs after which to model the behavior of a new IV, both semantic and syntactic information can be relevant; it is not clear which to prioritize. As a result, we study three different word embedding architectures, each of which encodes slightly different information: skip-gram, structured skip-gram, and CBOW. We also study the effects of different context sizes when training the embeddings. In section 3.2.1, we give an overview of how word vectors can be trained using these methods and how these vectors can be used to find similarity; sections 3.2.2 and 3.2.3 describe the differences between the architectures and the effects of context size, respectively.

3.2.1 Word embeddings

The goal of these word embedding architectures is to represent each word in the vocabulary using a continuous vector. Once we have these vector representations, we define *word similarity* between two words w_1 and w_2 as the cosine similarity between their vector representations $\mathbf{w_1}$ and $\mathbf{w_2}$:

$$sim(w_1, w_2) := \frac{\mathbf{w_1} \cdot \mathbf{w_2}}{\|\mathbf{w_1}\|\|\mathbf{w_2}\|} \tag{3.1}$$

Such vectors are learned directly from raw text based on distributional information. This is a similar process for each of the three architectures we study, and it is based on a shallow neural network architecture (Mikolov et al., 2013b).

3.2.2 Architectures

We study three architectures for creating word embeddings: skip-gram, CBOW, and structured skip-gram. In the skip-gram model, a word is input into the architecture, and the context words are predicted (Mikolov et al., 2013b). Words in both the left and right context are predicted; the number of words predicted is a hyperparameter (see section 3.2.3). The order of the words in the context is ignored. The CBOW model predicts a word given the words in its context (Mikolov et al., 2013b). Like for skip-gram, both the left and the right contexts are considered, and word order is ignored. Structured skip-gram is similar to skip-gram, but it takes into account word order. This is done by training the model to predict context words in specific positions relative to the input word, rather than just predicting which words are in the context (Ling et al., 2015).

The word vectors that result from each of these architectures encode both syntactic and semantic information. In fact, analysis by Mikolov et al. (2013b) suggested that skip-gram vectors performed better than CBOW vectors on semantic tasks, while CBOW vectors outperformed skip-gram vectors on syntactic tasks. However, Ling et al. (2015) showed that using skip-gram vectors yielded a higher accuracy than using CBOW vectors on their syntactic tasks. The structured skip-gram architecture was designed specifically to encode syntactic information by taking order into account, and as a result it does outperform both skip-gram and CBOW on syntactic tasks (Ling et al., 2015).

3.2.3 Context size

As described above, the word embedding architectures that we study either predict a word given the words in its context or predict the context given a word. We investigate the effects of varying the context size when training the word vectors. Note that *context size* refers to the number of words on either side of a given word in training. For example, if CBOW vectors are trained with a context size of two words, that means that for each word, the model takes as input the two words before and two words after it for a given occurrence in the training corpus and tries to predict the word.

We expect context size to have an effect on whether the word vectors encode more semantic or syntactic information. Indeed, Bansal, Gimpel, and Livescu (2014) found that using a smaller context size caused words with the same part of speech to be more similar, while using a larger context size caused words referring to the same topic to be more similar.

3.3 Language model estimation methods

In order to add new IVs to a bigram language model, four aspects must be addressed:

- Estimation of unigram probability of the new IVs ($P_{pre}(newIV)$)
- Estimation of backoff weights of the new IVs ($B(newIV)$)
- Finding bigrams containing new IVs
- Estimation of the probability of the new bigrams

We refer to the probability estimates as $P_{pre}(newIV)$ instead of $P(newIV)$. This is because adding probability mass causes the unigram and bigram probabilities to not be normalized; hence, the values assigned for these probability estimates are not the final probabilities assigned. Unigram and bigram probabilities are renormalized after estimation; see section 3.4 for a specification of how this is done. In addition, many of the new probability estimates are based on probabilities of existing n-grams in the original language model; such probabilities will be denoted as $P_{orig}(x)$ and $P_{orig}(y|x)$[1].

Note further that a bigram containing a new IV could have the new IV as the first word in the bigram (bigrams of the form *newIV-x*) or as the second word in the bigram (bigrams of the form *x-newIV*). The preliminary probability estimations of these bigrams are denoted as $P_{pre}(x|newIV)$ and $P_{pre}(newIV|x)$, respectively. We propose three sets of approaches for estimating these language model parameters.

Similarity-based approaches take advantage of word similarity measures to find similar IVs to a new IV; the behavior of the new IV is then modeled after the behavior of the similar IVs. In practice, we use the similarity measures described in section 3.2, although many different measures could be applied in our methods.

Corpus-based approaches use the contemporary corpus to inform language model estimation. The behavior of the new IVs is based on their behavior in the contemporary corpus. Note that corpus-based approaches also use word similarity, although the basis of these approaches

[1]We emphasize that P_{pre} is not a probability function, while P_{orig} is. By renormalizing the unigram and bigram values of P_{pre} as described in section 3.4, we create a probability function P.

is the contemporary corpus. Similarity-based approaches, on the other hand, do not use any outside corpus at language model estimation time (although outside data is used to create the pre-trained word similarity measure).

Mixed approaches combine the techniques used in both similarity-based approaches and corpus-based approaches for finding bigrams containing new IVs and estimating bigram probabilities. These approaches attempt to take advantage of all of the available information to improve bigram estimation.

3.3.1 Similarity-based approaches

This section discusses language model estimation approaches that rely only on the behavior of similar words; no contemporary corpus is used. This is based on the hypothesis that a new IV will behave more or less like the existing IVs that are most similar to it; here, we call such words *similar IVs*. Hence, we use the probabilities of n-grams containing those words to estimate the probabilities of n-grams containing the new IV.

The advantage of our similarity-based approaches is that no contemporary corpus for finding word contexts is required. However, the new IVs are likely to be rare words, particularly in the pre-trained data (since they are mostly proper names that have become relevant only recently). This means that it can be difficult to establish reliable word similarity measures between them and the IV words.

All of the proposed similarity-based approaches use the same overall process to estimate the language model parameters. For each new IV, this process is as follows:

1. Find the similar IVs to the new IV
2. Use the unigram probabilities of the similar IVs to estimate $P_{pre}(newIV)$
3. Use the backoff weights of the similar IVs to estimate $B(newIV)$
4. Choose a similar IV to use for new IV bigram estimation
5. Find all bigrams containing the similar IV
6. In some cases, select from among those bigrams (otherwise, take all such bigrams)
7. Add the new bigrams to the language model with the same probabilities, replacing the similar IV in the bigram with the new IV

Once each new IV is added to the language model, the model is renormalized. In the rest of this section, we will specify our variations on this approach.

Unigram probability estimation

One method for estimating $P_{pre}(newIV)$ using similar IVs is to take the most similar IV (called the *closest IV*) and assign its probability to the new IV unigram, i.e., $P_{pre}(newIV) \leftarrow P_{orig}(closestIV)$. However, the word embeddings are somewhat unreliable for rare words (which the new IVs often are), so using only the closest IV can be risky. In addition, the closest IV to a new IV is often a rare word itself, so this could have the effect of giving too little probability to the new IV. As a result, we also study methods that use the N most similar IVs to estimate the new IV's unigram probability.

We propose the following ways of using the probabilities of the N most similar IVs to estimate $P_{pre}(newIV)$, where $sim(simIV, newIV)$ is the word similarity between a similar IV and the new IV:

- Maximum: $P_{pre}(newIV) \leftarrow \max_{simIVs} P_{orig}(simIV)$

- Minimum: $P_{pre}(newIV) \leftarrow \min_{simIVs} P_{orig}(simIV)$
- Median: $P_{pre}(newIV)$ is median probability of similar IVs
- Average: $P_{pre}(newIV) \leftarrow \frac{1}{N} \sum_{simIVs} P_{orig}(simIV)$
- Average weighted by word similarity:
  $P_{pre}(newIV) \leftarrow \frac{1}{N} \sum_{simIVs} P_{orig}(simIV) \times \max(0, sim(simIV, newIV))$
- Average weighted by percent of maximum word similarity: average weighted by similarity, multiplied by $\frac{1}{\max_{simIVs} sim(simIV, newIV)}$ iff $\max_{simIVs} sim(simIV, newIV)$ is positive

Note that since we use cosine similarity as our word similarity measure, the maximum value for word similarity is 1. Hence, weighting the mean by word similarity has the effect of lowering the actual mean; therefore, we study weighting by percent of maximum similarity as well in order to lessen this effect. In addition, if either of the weighted average methods would result in a zero (or negative) probability being given to a new IV, we instead assign the minimum of the similar IV probabilities to that new IV.

In the cases where we take the minimum or maximum probability, there is a single similar IV that is used to give its probability to the new IV. We refer to this similar IV as the *IV used*.

Estimation of backoff weights

Since the original language model uses modified Kneser-Ney smoothing, the original LM backoff weight for a given word depends on its bigrams. As a result, for estimating backoff weight for a new IV, we simply use the backoff weight of the IV used to find the new IV's bigrams (see below for how this IV is chosen). In addition, only the backoff weights of new IVs are changed; no existing backoff weights are modified.

Finding bigrams

Our initial strategy for finding bigrams containing the new IV is to retrieve all bigrams containing the closest IV, replace the closest IV with the new IV, and add the new bigrams to the model. This method is outlined in pseudocode in algorithm 1. In cases where the unigram

Algorithm 1 Find bigrams using closest IV.

```
 1: procedure ADDBIGRAMS(newIVs)
 2:     newBigrams = {}
 3:     for newIV in newIVs do
 4:         closestIV = findClosestIV(newIV)                    ▷ find closest IV
 5:         prob = getProb(closestIV)          ▷ unigram probability and backoff
 6:         backoff = getBackoff(closestIV)
 7:         push(Unigrams, (newIV, prob, backoff))
 8:         closestIVBigrams = getBigrams(closestIV)
 9:         for bigram in closestIVBigrams do
10:             newBigrams += replace(bigram, closestIV, newIV)
                                                               ▷ find bigrams
11:     Bigrams += newBigrams
```

probability of the new IV is defined using the maximum or minimum probability of the similar IVs, we also consider a variation of this method in which we use the bigrams of the IV used to define the unigram probability (rather than the closest IV).

In order to avoid adding too much noise to the adapted language model, we also propose limiting the number of bigrams added for each new IV. For each new IV, we choose up to M bigrams to add to the language model; if M bigrams were not found for that new IV, we add all of the bigrams available. We study the following ways of choosing the bigrams:

- M bigrams with the highest probability
- M bigrams with the highest probability relative to the total probability of all bigrams beginning with the same word (note that the total probability is not necessarily 1, due to backoff weights)
- M bigrams with the highest probability relative to the maximum probability of all bigrams beginning with the same word
- $\frac{M}{2}$ x-newIV bigrams with the highest probability, and $\frac{M}{2}$ newIV-x bigrams with the highest probability

Note that, as illustrated in algorithm 1, the bigrams added to the language model are kept separate from the bigrams in the original language model until all bigrams are found. This means that new bigrams are not added dynamically, so all new bigrams contain exactly one IV and one new IV. No bigrams containing two new IVs are added to the language model in the similarity-based approaches.

Bigram probability estimation

Probabilities of new bigrams are assigned as follows. Each new bigram is based on a unique existing bigram (called the *corresponding bigram*), in which the new IV replaced an IV. Hence, the new bigram is assigned the probability of the corresponding bigram, and the probabilities are renormalized. Note that in the cases where only M bigrams are added to the language model for each new IV, the probabilities may be different for the new bigram and its corresponding bigram after renormalization.

3.3.2 Corpus-based approaches

The data-driven word representations that similarity-based approaches to LM estimation are based on may be unreliable for the new IVs, since they are likely to be relatively rare in the corpus used to estimate word vectors. Therefore, we propose another set of strategies for language model estimation, this time based on the contemporary corpus. These corpus-based approaches have the advantage of using temporally relevant data to adapt the language model; however, unlike the similarity-based approaches, they require the use of additional data. Note that corpus-based approaches use word similarity as well, but the main source of information for updating the language model is the contemporary corpus.

The overall process for estimating language model parameters using a corpus-based approach is as follows:

1. Calculate statistics on new IV occurrences in the contemporary corpus
2. Extract bigrams containing new IVs from the contemporary corpus
3. For each new IV:
   (a) Find the similar IVs
   (b) Use the corpus statistics (possibly combined with similar IV probabilities) to estimate $P_{pre}(newIV)$

   (c) Use information about the similar IVs to estimate $B(newIV)$
   (d) Decide which of the contemporary corpus bigrams containing the new IV to add to the language model
   (e) Estimate bigram probabilities using information about similar IVs and corpus statistics

4. Renormalize the language model

Unigram probability estimation

In our experiments, we adapt the baseline language model rather than the original language model[2]; an important consequence of this strategy is that all of the new IVs have an initial (very small) probability in the unadapted language model. Therefore, we propose using information about the behavior of the new IVs in the contemporary corpus to inflate the baseline probabilities of the new IVs ($P_{base}(newIV)$). We also consider a method that combines contemporary corpus statistics with word similarity measures.

Specifically, we study four ways of defining unigram probabilities for the new IVs, where $N(x)$ is the count of x in the contemporary corpus and $K := max_{newIVs}N(newIV)$:

- Maximum likelihood:
  $P_{pre}(newIV) \leftarrow \max\left(P_{base}(newIV), \frac{N(newIV)}{\sum_{w \in corpus} N(w)}\right)$
- Weighted by corpus occurrences[3]:
  $P_{pre}(newIV) \leftarrow P_{base}(newIV) \times (1 + N(newIV))$
- Scaled closest IV probability:
  $P_{pre}(newIV) \leftarrow P_{orig}(closestIV) \times \frac{1+N(newIV)}{1+K}$
- Scaled maximum probability of similar IVs:
  $P_{pre}(newIV) \leftarrow \max_{simIVs} P_{orig}(simIV) \times \frac{1+N(newIV)}{1+K}$

Note that when weighting by counts, we augment the counts by 1, since not all new IVs are in the contemporary corpus (and we do not want to give a zero probability to any new IVs). In addition, when weighting by corpus occurrences, we only scale the weights when we are weighting similar IV probabilities (not when weighting the baseline probabilities). This is because we expect the similar IV probabilities to be significantly higher than the baseline new IV probabilities, so scaling the probabilities allows us to avoid giving too much probability mass to the new IVs.

Estimation of backoff weights

Backoff weights of the new IVs ($B(newIV)$) are assigned based on the backoff weights of the similar IVs or of <unk>; we do not attempt to train backoff weights using the corpus. The motivation for doing so is that we expect overall behavior of the new IVs to be similar to that of the similar IVs or of <unk>, whereas training reliable Kneser-Ney backoff weights from the corpus would likely require much more information about the new IVs.

We study the following methods for assigning backoff weights:

[2]The baseline LM is an modification of the original LM where a small dummy probability has been given to each new IV. We adapt the baseline model to ensure that all adapted LMs have the same vocabulary, in order to allow for direct perplexity comparisons. See section 4.4 for details on how this model was created.

[3]Weighting prior probabilities by raw corpus occurrences makes sense given our experimental setup (see chapter 4); in particular, a small contemporary corpus is used and the baseline probability is negligible, so the risk of assigning too high of probability to a unigram is low. However, in different experimental setups, it might be preferable to scale these weights in some way.

- <unk> backoff: $B(newIV) \leftarrow B(\text{<unk>})$
- Closest IV backoff: $B(newIV) \leftarrow B(closestIV)$
- Maximum backoff of N similar IVs such that the backoff is not 1:
  $B(newIV) \leftarrow max_{simIVs:B(simIV)\neq 1} B(simIV)$
- Minimum backoff of N similar IVs:
  $B(newIV) \leftarrow min_{simIVs} B(simIV)$

In order for bigrams of the form newIV-x to be added to the language model, $B(newIV)$ should not be 1. As a result, if taking either the minimum or the maximum of similar IV backoff weights results in a backoff weight of 1, we take $B(\text{<unk>})$ instead. On the other hand, if there are no bigrams of the form newIV-x in the language model for a given new IV, then the new IV must be given a backoff weight of 1. As a result, after assigning backoff weights and adding bigrams to the LM, we change backoff weights to 1 for all such new IVs.

Note that only backoff weights of new IVs were changed. No existing backoff weights were modified, and unigrams that had no backoff weights were not assigned a new backoff weight.

Finding bigrams

One of the major advantages of having access to the contemporary corpus is that the corpus can be used to find bigrams. Our initial proposal for doing this is as follows: for each new IV, add a bigram containing that new IV if and only if that bigram was found in the corpus. These bigrams can be interpreted as safe choices: since they actually occur in the contemporary corpus, we can be relatively confident that they will occur in the test data.

Note that when a bigram containing a new IV and an OOV is found in the contexts corpus, it is ignored, since we cannot add bigrams containing OOVs to the language model. However, it is possible for the corpus to contain a bigram consisting of two new IVs; we call these *newIV-newIV* bigrams. Indeed, we do not expect this to be an edge case, since many of the new IVs are names of people; the first and last names of a person often appear next to each other and are often both new IVs. We study two options for addressing these bigrams: ignoring them or adding them to the language model. We take care to add each newIV-newIV bigram only once to the language model (see line 13 of algorithm 2).

We consider an extension to the basic approach of adding all bigrams from the contemporary corpus. This consists of implementing a cutoff for bigram occurrences; only bigrams that occur more than the cutoff amount of times in the contemporary corpus are added to the language model. This method allows us to add only bigrams about which we are quite confident; although the coverage of the bigrams might not be as complete, we expect the resulting new bigrams to be more likely to appear in the test article. We refer to this approach as the *limiting bigrams approach*.

On the other hand, we propose a strategy to artificially generate bigrams to be added to the LM that did not occur in the contemporary corpus. We refer to this strategy as the *generating bigrams approach*. First, we find all new bigrams to be added to the language model from the contemporary corpus. We then generate additional bigrams for each new bigram as follows, where x refers to the IV word in the bigram:

1. Find all IVs y such that $sim(x, y)$ is greater than a cutoff
2. For each y:
   (a) Create an additional bigram based on the new bigram, replacing x with y
   (b) Add the additional bigram to the language model

Algorithm 2 Find bigrams using corpus contexts.

```
1: procedure FINDBIGRAMS(newIVs)
2:     newBigramsWord1 = {}                                  ▷ bigrams of form newIV-x
3:     newBigramsWord2 = {}                                  ▷ bigrams of form x-newIV
4:     for newIV in newIVs do
5:         wordsAfter = findAfter(newIV, corpus)             ▷ bigrams newIV-x
6:         for word in wordsAfter do
7:             push(newBigramsWord1, (newIV, word))
8:             closeIVs = findCloseIVs(word, minSim)
9:             for closeIV in closeIVs do
10:                push(newBigramsWord1, (newIV, closeIV))
11:        wordsBefore = findBefore(newIV, corpus)           ▷ bigrams x-newIV
12:        for word in wordsBefore do
13:            if word not in newIVs then                    ▷ already added newIV-newIV
14:                push(newBigramsWord2, (word, newIV))
15:                closeIVs = findCloseIVs(word, minSim)
16:                for closeIV in closeIVs do
17:                    push(newBigramsWord2, (closeIV, newIV))
18:        Bigrams += newBigramsWord1 + newBigramsWord2
```

This process is specified in pseudocode in algorithm 2.

This artificial bigram generation method allows us to increase the coverage of the model, which may be beneficial when the contemporary corpus is very small. Note that in order to avoid adding too much noise, we only create additional bigrams based on new bigrams where one word is an IV. This means that no additional bigrams are generated based on newIV-newIV bigrams. Our main reason for adding newIV-newIV bigrams was to include bigrams consisting of a person's first and last names, so it would not make sense to augment these using similar IVs.

Bigram probability estimation

There are two types of new IV bigram probabilities to consider: probabilities of the form $P_{pre}(x|newIV)$ and probabilities of the form $P_{pre}(newIV|x)$. We consider newIV-newIV bigrams as part of the first case (bigrams of the form newIV-x); in the other case, probability estimation depends on the probabilities of existing bigrams starting with the same first word, which are not reliable when the first word is a new IV.

$P_{pre}(x|newIV)$ bigram probabilities are estimated based on corpus statistics and word similarity. We propose the following methods for estimating such probabilities, where $N(newIV\text{-}x)$ is the count of bigram *newIV-x* in the contemporary corpus.

- Uniform probability: $P_{pre}(x|newIV) \leftarrow 1$
- Weighted by similarity:
  - If bigram newIV-x was added to the LM directly from the corpus: $P_{pre}(x|newIV) \leftarrow 1$
  - If bigram newIV-x was added through the generating bigrams approach based on a bigram newIV-y[4]: $P_{pre}(x|newIV) \leftarrow sim(x, y)$

[4]Recall that we use cosine similarity as our word similarity measure, so $sim(x, y) \leq 1$ for all x and y. In addition, in the generating bigrams approach, we use a similarity cutoff to create the bigram newIV-y, so $sim(x, y) \geq$ cutoff for all x and y. We take care to take a similarity cutoff greater than zero so that $P(x|newIV)$ will always be positive.

- Weighted by number of occurrences in the contemporary corpus:
  $P_{pre}(x|newIV) \leftarrow 1 + N(newIV\text{-}x)$

Weighting the probability by similarity is only applicable to the generating bigrams approach to finding bigrams. Note that when weighting by similarity, multiple weights could be given to the same bigram; in this case, we take the maximum weight. The weighting methods allow us to give a higher weight to bigrams that we are more confident about (either because they occur in the corpus or because they occur more often in the corpus).

$P_{pre}(newIV|x)$ bigram probabilities can be estimated using information about the behavior of existing bigrams with the same first word, *x*. We will refer to such existing bigrams as bigrams of the type *x*-*y* (where *y* is an IV word), and to their probabilities as $P_{orig}(y|x)$. The following approaches to estimating $P_{pre}(newIV|x)$ are studied:

- Minimum: $P_{pre}(newIV|x) \leftarrow \min_{y:P_{orig}(y|x)\neq 0} P_{orig}(y|x)$
- Maximum: $P_{pre}(newIV|x) \leftarrow \max_{y:P_{orig}(y|x)\neq 0} P_{orig}(y|x)$
- Probability of closest IV with non-zero bigram probability:
  $P_{pre}(newIV|x) \leftarrow P_{orig}(\text{argmax}_{y:P_{orig}(y|x)\neq 0}\, sim(y, newIV)|x)$
- Max. probability of most similar IVs with non-zero bigram probability:
  $P_{pre}(newIV|x) \leftarrow max_{simIVs:P_{orig}(simIV|x)\neq 0} P_{orig}(simIV|x)$

In the fourth case, N similar IVs such that $P_{orig}(simIV|x) \neq 0$ are taken. These four strategies allow us to study the trade-off between giving a high probability to bigrams containing the new IVs (since we are relatively confident that they will occur) and giving a low probability to bigrams containing the new IVs (to avoid taking too much probability from existing, common sequences in the LM).

3.3.3 Mixed approaches

It is possible to combine the similarity-based approaches specified in section 3.3.1 with the corpus-based approaches specified in section 3.3.2. In particular, the mixed approaches we study consist of using both of these approaches to increase the bigram coverage of the language model.

For finding bigrams, the corpus-based approaches have the disadvantage that we only add bigrams for new IVs that actually occur in the contemporary corpus. Furthermore, in the case of the limiting bigrams approach for finding bigrams, some new IVs that occur in the corpus may not have any bigrams added (i.e., if they do not have any bigrams that occur more than the cutoff amount of times). The mixed approach attemps to remedy this as follows:

1. Use one of the corpus-based approaches to add bigrams containing new IVs to the language model
2. For each new IV that does not have any bigrams in the LM, use one of the similarity-based approaches to generate bigrams for that IV and add them to the model

3.4 Language model renormalization

Since the language model estimation methods described in section 3.3 involve adding n-grams and probabilities to a language model, the models that result from these algorithms are not normalized. Therefore, after the probabilities are estimated, the model must be renormalized so that it represents a true probability distribution. In this section, we describe the method we use to renormalize the bigram language models. We will refer to this as *standard renormalization*; see section 3.5.2 for a discussion of issues with standard renormalization and suggestions for improvements.

In standard renormalization, the probability mass given to new n-grams is taken from each existing n-gram proportionally to its original probability. Note that the backoff weights are not affected.

In order to renormalize the probabilities, the constraint that all the probabilities must sum to 1 is used. Taking into account backoff weights, the unigram and bigram constraints can be expressed as follows, for all words w_1 in the vocabulary:

$$\sum_{w} P(w) = 1 \tag{3.2}$$

$$\sum_{\{w_2|P(w_2|w_1)\neq 0\}} P(w_2|w_1) = 1 - B(w_1) + B(w_1) \times \sum_{\{w_2|P(w_2|w_1)\neq 0\}} P(w_2) \tag{3.3}$$

where $P(w)$ is the language model probability of the unigram w, $P(w_2|w_1)$ is the LM probability of the word w_2 given the history w_1, and $B(w)$ is the backoff weight of the word w. Note that equation (3.3) is a rewritten version of the standard renormalization equation that uses equation (3.4) to speed up calculations.

$$\sum_{\{w_2|P(w_2|w_1)\neq 0\}} P(w_2) = 1 - \sum_{\{w_2|P(w_2|w_1)=0\}} P(w_2) \tag{3.4}$$

Thus, standard renormalization of unigrams is quite simple. It is done by summing the probabilities of the unnormalized model and dividing each unigram probability by that sum. Bigram probabilities $P(w_2|w_1)$ have to be renormalized separately for each w_1 by multiplying each bigram probability by the right-hand side of equation (3.3) and dividing by the left-hand side of the same equation. Note that since bigram probabilities depend on unigram probabilities, unigrams must be renormalized before renormalization of bigrams can occur.

3.5 Additional strategies

The previous sections in this chapter covered adaptation strategies that are implemented in this thesis. In this section, we describe some additional methods and extensions that could be used for dynamically adding OOVs to a language model. These have not been implemented and should be considered as options for future work.

3.5.1 Higher-order n-grams

Since our ASR system uses a bigram language model to calculate first-pass probabilities, we have concentrated on parameter estimations for bigram LMs in our experiments. However, it is possible to extend these methods to trigrams or higher orders; in this section, we will describe how to do so. In order to update a trigram language model, the additional elements that need to be addressed are finding trigrams, estimating trigram probabilities, assigning bigram backoff weights, and renormalizing.

Finding trigrams

Note that there are three different types of trigrams that could be added to the language model, of the following forms: x-y-newIV, x-newIV-y, and newIV-x-y. Our proposed strategies for finding bigrams extend easily to the trigram case, with only trivial modifications. It should be mentioned, though, that data sparsity is likely to be a bigger problem when finding trigrams using corpus-based approaches, so generating trigrams is likely to be more important than for the bigram case. When generating artificial trigrams, there is an additional question to study: should we allow both of the context words to be replaced with a similar IV at the same time? This would create many more trigrams, but it could also create more noise.

One important constraint to keep in mind is that if a trigram x-y-z is added to the language model, the bigram x-y must also be in the language model. For most of our proposed strategies for finding n-grams, this happens automatically. The exceptions are generating trigrams and restricting the number of trigrams added per new IV. There are two options that could be studied here: do not add any trigram to the LM if the prefix bigram is not in the LM, or add the prefix bigram to the LM (if it is not present) when the trigram is added. In the second case, we could use similar methods to those proposed in section 3.3 to assign probabilities to these new bigrams.

Trigram probability estimation

We can estimate trigram probabilities using similar techniques to those that were proposed for the bigram case. Strategies for estimating probabilities of trigrams of the form newIV-x-y or x-newIV-y can be based on our methods for estimating $P(x|newIV)$. Probabilities of trigrams of the form x-y-newIV can be estimated similarly to our methods for estimating $P(newIV|x)$.

Estimation of bigram backoff weights

For estimating unigram backoff, the methods we have proposed have been to use the backoff weight of the closest IV word or to choose from among the backoff weights of the N closest IV words by taking the maximum or minimum. Bigram backoff could work similarly; for a bigram of the form x-newIV, we would take the backoff weight of the bigram x-y, where y is the closest IV to the new IV such that the bigram has a backoff weight in the model (and similarly for bigrams of the form newIV-x).

Renormalization

Standard renormalization for a trigram language model is a simple extension of the bigram case explained in section 3.4. In addition to the constraints in equations (3.2) and (3.3), the following trigram constraint must be fulfilled:

$$\sum_{\{w_3|P(w_3|w_1,w_2)\neq 0\}} P(w_3|w_1,w_2) = 1 - B(w_1w_2) + B(w_1w_2) \times \sum_{\{w_3|P(w_3|w_1,w_2)\neq 0\}} P(w_3|w_2) \quad (3.5)$$

where $P(w_3|w_1,w_2)$ is the probability of the word w_3 given the history w_1w_2 and $B(w_1w_2)$ is the backoff weight of the bigram w_1w_2. Once unigrams and bigrams are renormalized, trigram probabilities $P(w_3|w_1,w_2)$ can be renormalized separately for each prefix w_1w_2 by multiplying each trigram probability by the right-hand side of equation (3.5) and dividing by the left-hand side of the same equation.

3.5.2 Different renormalization techniques

The standard renormalization procedure described in section 3.4 has the advantage of being relatively quick to implement and simple to understand. However, taking probability mass from all n-grams proportionally may not make intuitive sense. For example, since the new IVs in our experiments are all proper nouns, taking more probability mass only from existing proper nouns might be an improvement over the standard method.

In this section, we describe one simple alternative to standard renormalization that does not require any additional training data or modifications to our data. The intuition motivating this method is that word class distributions should not change greatly in texts from different time periods. This hypothesis should be tested experimentally before we study this renormalization method further.

This approach consists of normalizing the model by classes, so that probability mass remains constant across different word classes. However, in order to avoid having to train additional word classes, we would classify all IV words as their own class, and put the new IVs in a class with <unk>. Thus, unigram probabilities for IV words would stay the same, and the extra unigram probability mass would be taken from <unk> and from each new IV proportionally to its probability. For bigrams in which the first word is a new IV, standard renormalization would be used, since there is no prior knowledge of a probability distribution over classes for such bigrams. For other bigrams, renormalization would work similarly to unigram renormalization; the probability mass given to a bigram x-<unk> in the original language model should be equal to the sum of the probability masses given to all bigrams of the form x-<unk> or x-newIV in the new language model. If the bigram x-<unk> did not exist in the original language model, standard renormalization can be used instead in order to avoid giving zero probability to bigrams added to the model.

Another option would be to use a class-based normalization as described above, but create different word classes (rather than putting all new IVs into the same class). Such classes could be trained relatively easily from the word embeddings. It might also make sense to change the backoff weights when renormalizing. Since the original language model was trained using modified Kneser-Ney smoothing, the backoff weight of a word roughly corresponds to how likely it is for that word to be followed by <unk>. Hence, since the number of unknown words diminishes when we add new IVs to the models, it is possible that reducing backoff weights could be beneficial.

3.5.3 Using language models to estimate similarity

Many of the methods described throughout this chapter have relied on word similarity. In our experiments, we use cosine similarity between word embeddings, as described in section 3.2. In this section, we propose an alternative way of finding similar words to a new IV based on perplexity.

This method relies on two pieces of input: the contemporary corpus and a language model (such as the 4-gram or recurrent neural network language models that are used in the second pass of our ASR system). It should be noted that low-order LMs (such as bigram models) are probably not suited to this method.

We propose using language model perplexity to find the most similar IV(s) to a given new IV. This would be done as follows, for a given new IV:

1. Create a subcorpus from the contemporary corpus consisting of each sentence containing the new IV
2. For each word x in the language model vocabulary:
   (a) Replace all occurrences of the new IV in the subcorpus with x
   (b) Calculate the perplexity of the LM on the modified subcorpus
3. Rank the words in the vocabulary by perplexity; the word that gives the lowest perplexity is most similar to the new IV

Note that while this method gives a complete ranking of closeness to the new IV for all words in the vocabulary of the original language model, it does not assign a meaningful similarity score between two words (like cosine similarity when using word embeddings). In addition, the rankings are not symmetric, and there is no way of finding similarity between two new IVs. These are clear limitations of the model; however, we do not use this information in most of our proposed methods. In addition, like with word embeddings but unlike with some other word similarity methods, no inference can be made about new IVs that do not appear in the corpus at all.

One major feature of this similarity model that distinguishes it from word embeddings is that it has a strong bias towards words with higher probabilities in the language model; that is, such words are more likely to have higher similarity rankings with all words. This is quite different from similarity using word embeddings, where rare words (as many new IVs are) tend to have a higher similarity to other rare words. It is not clear whether this feature is advantageous or disadvantageous. On the one hand, new IVs being judged as most similar to common words would mean that a relatively high probability is given to these new IVs, which may be counter-intuitive to the fact that the new IVs are often proper nouns that are not particularly common. However, it might be better to use more common IVs when depending on the IVs to add bigrams to the model. This is because the more common words will be better modeled in the LM, due to the fact that there were more examples of them in the training data (so there are less likely to be anomalies). In any case, it is possible to normalize the rankings given by perplexity in order to reduce the bias towards high-probability words. For example, we could divide the resulting perplexity by the unigram probability of the IV in the language model.

Chapter 4

Experimental Setup

In this chapter, we give an overview of the experimental framework used to evaluate the methods introduced in chapter 3. This includes information about data, language models used, and evaluation. In chapter 5, we will present and analyze the results of these experiments.

4.1 Data

In our experiments, we use corpora for the following purposes:

- Training corpora
  - Original language model
  - Word embeddings
  - New IV lists
- Contemporary corpus for corpus-based approaches (see section 3.3.2)
- Development corpus
- Test corpus

This section describes each of the corpora used for these purposes.

4.1.1 Training corpora

Training data is required to create the original language model (see section 4.4), the word embeddings (see section 4.3), and the new IV lists (see section 4.2). For each of these tasks, some combination of the following three training sets is used:

- *Le Monde* + Gigaword: textual data from the French newspaper *Le Monde* and from the French Gigaword corpus; news articles published between 1994 and 2008
- *Le Figaro*: textual data from the French newspaper *Le Figaro*; news articles published in 2014
- *L'Express*: textual data from the French newspaper *L'Express*; news articles published in 2014

Table 4.1 gives an overview of the sizes of each of these corpora.

The original language model is trained using the *Le Monde* + Gigaword corpus, which is mismatched with the time period of the test data (see section 4.1.4). The word embeddings are created using a concatenation of the three corpora, which allows them to include some data from the same time period as the test corpus. Finally, the new IV lists are created using the *L'Express* corpus; see section 4.2 for information on this process.

Corpus	**Sentences**	**Word tokens**
Le Monde + Gigaword	40,500,344	1,075,341,501
Le Figaro	354,543	7,970,360
L'Express	2,049,597	51,042,525
TOTAL	42,904,484	1,134,354,386

TABLE 4.1: Sizes of training corpora.

4.1.2 Contemporary corpus

The corpus-based language model estimation strategies described in section 3.3.2 make use of a contemporary corpus that contains textual data from the same time period as the test data. We use the *Le Figaro* corpus described in section 4.1.1 for this purpose.

Despite the small size of the contemporary corpus, most of the new IVs are represented in it. Table 4.2 shows the percentage of the new IVs from the whole-test new IV list (see section 4.2) that occur at least once, at least twice, and at least five times in the contemporary corpus. For reference, there is a total of 6,382 new IVs in this list; these represent the new IVs to be added to the language models for the test data.

Occurrences	**Percent of new IVs**
$\geq$ 1 time	64.7
$\geq$ 2 times	51.4
$\geq$ 5 times	31.9

TABLE 4.2: Occurrences of new IVs in the contemporary corpus.

4.1.3 Development corpus

The development corpus comes from the website of the news channel Euronews[1]. It consists of 1,962 textual news articles from January 2014 to June 2014. This corpus is used to evaluate each of our proposed approaches and select the approaches for which to run the test perplexity and recognition experiments. The experiments on the development corpus are described in section 5.2.

Table 4.3 shows statistics about the size of the development corpus, as well as the average size of the development articles. Note that OOV counts are with respect to the original language model vocabulary, and include all OOVs in the data. This means that they are not restricted to only new IVs or only OOVs with word vectors. In addition, some articles in the data did not contain any OOVs; no distinction is made (here or in the experiments) between articles without OOVs and those with OOVs.

	Sentences	**Word tokens**	**OOV tokens**
whole corpus	24,040	510,351	13,768
average per article	12.3	260.1	7.0

TABLE 4.3: Development corpus total size and average article size.

[1]http://fr.euronews.com/

4.1.4 Test corpus

The development corpus described in the previous section is used to tune the parameters and study the performance of all of our proposed algorithms. Once we have found the best parameter configurations and algorithms on the development corpus, we use the test corpus to further examine their performance.

Like the development corpus, the test corpus is also from the Euronews website; in this case, however, the data consists of video reports and their accompanying transcripts. The video reports are used for the recognition experiments, while the transcripts are used for the perplexity experiments. It is important to note that the reference transcriptions for the recognition experiments are the transcripts provided with the news videos, which may not always be an exact match to the audio. The articles were published in the first half of 2014; hence, they are from the same period as the development data (see section 4.1.3) and the contemporary corpus (see section 4.1.2), but from a different time period from the training data for the original language model (see section 4.1.1).

The test corpus consists of 467 articles; these were selected by randomly choosing four weeks from January to June of 2014 and including all articles from those weeks. The weeks chosen were January 17–23, 2014 (73 articles); February 14–20, 2014 (125 articles); March 9–15, 2014 (132 articles); and April 20–26, 2014 (137 articles). We use only articles from the selected weeks because we create separate adapted language models for each article in the test data (unlike for the development data). This would have been too computationally expensive to do for more articles.

Table 4.4 shows statistics about the size of the test corpus, as well as the average size of the test articles. As in table 4.3, the OOV counts include all OOVs in the data with respect to the vocabulary of the original LM. In addition, some articles in the data did not contain any OOVs; no distinction is made (here or in the experiments) between articles without OOVs and those with OOVs. Note that the articles in the test data are slightly shorter on average than the articles in the development data (see table 4.3).

	Sentences	**Word tokens**	**OOV tokens**
whole corpus	4,689	91,880	2,074
average per article	10.1	196.7	4.4

TABLE 4.4: Test corpus total size and average article size.

4.2 New IV lists

For each article, a list of new IVs to be added to the article (called the *new IV list*) is generated. These lists are derived from rankings of OOVs for each article, which are provided to us by Imran Sheikh.

The OOV rankings are created using the NBOW2 model introduced by Sheikh et al. (2016b). This is done as follows:

1. The list of OOV proper nouns (with respect to the original language model) is extracted from the *L'Express* corpus described in section 4.1.1
2. The *L'Express* corpus is also the basis for creating a vector space using the NBOW2 model
3. For each article:
   (a) The recognition system using the original language model is run on the article to create an initial recognition hypothesis

(b) The initial hypothesis is projected into the vector space

(c) The proper noun OOVs are ranked based on their closeness to the article in the vector space

Note that the OOVs in this model are all proper nouns that are in the *L'Express* corpus but not in the original language model. The NBOW2 algorithm thus generates a ranking of words for each article by how likely they are to appear in that article.

From the rankings of OOVs for each article that are provided to us, we use the 128 words with the highest rankings to create the per-article new IV lists. Note that we only retain new IVs in the new IV lists if we have an associated word vector for them (see section 4.3); excluding words without word vectors results in an average of 119.3 new IVs on the new IV list per development article.

We use an automatic grapheme-to-phoneme conversion algorithm to assign pronunciations to the new IVs. This is based on conditional random fields (CRF); further details can be found in Illina, Fohr, and Jouvet (2011).

Note that a separate new IV list is created for each article in both the development and the test data. In addition, we create a *whole-development new IV list* and a *whole-test new IV list* by combining the new IV lists for all articles in the development and test corpora, respectively. The whole-development new IV list contains 8,066 new IVs, while the whole-test new IV list contains 6,382 new IVs.

Table 4.5 shows the average type- and token-level OOV recall when the top 32, 64, 128, and 256 words from the NBOW2-generated lists are taken for each article in the development corpus. As can be seen from the table, adding the top 128 words gives an amount of coverage of the OOVs that is not improved upon at all by taking the top 256 words.

List size	**Token recall (%)**	**Type recall (%)**
32 words	28.2	25.4
64 words	29.5	27.3
128 words	30.6	28.9
256 words	30.6	28.9

TABLE 4.5: Average recall on articles in the development corpus for per-article new IV lists of different sizes.

4.3 Continuous word representations

The training corpus used to create the word embeddings was described in section 4.1.1. Only those words that occur at least five times in the training corpus are counted in the vocabulary, resulting in a vocabulary size of 463,890 words. We use both the word2vec (Mikolov et al., 2013b) and the wang2vec (Ling et al., 2015) toolkits to train the word embeddings.

As described in section 3.2, we study the effects of using skip-gram, CBOW, and structured skip-gram models for our adaptation methods. We also investigate different context sizes for the skip-gram and CBOW algorithms. The context sizes examined are fifteen, ten, five, and two words. For the structured skip-gram model, we use only a context size of five words, following what was done by Ling et al. (2015).

In section 5.1, we describe the comparison experiments for these word embeddings. In subsequent experiments, we use the embeddings that yield the lowest perplexity in the comparison experiments.

4.4 Language models

We use the original language model of the system, a baseline model, and an oracle model in our experiments. In this section, we give a description of each of these models and how they are created. Note that separate baseline and oracle models are trained for the development data and for the test data. Here, we describe the models for the test data. The models for the development data are created in the same way as described below; the only difference is that the whole-development new IV list is used to create those models, rather than the whole-test new IV list (see section 4.2).

The original language model to be adapted is the same for all experiments. It is a bigram model with modified Kneser-Ney smoothing trained on the *Le Monde* and Gigaword data described in section 4.1.1. Both the original language model and the lexicon were provided for this work by Dominique Fohr and Irina Illina; these were part of our original ASR system.

Since we add words to the vocabularies of the adapted models, these models do not have the same vocabulary as the original language model. This means that we cannot compare the perplexities of these models with the perplexity of the original language model. As a result, we create the baseline language model, which has the same vocabulary as the adapted models.

One baseline language model is trained for the entire test corpus. It is created from the original language model by adding the whole-test new IV list to the language model vocabulary. Only unigrams corresponding to the new IVs are added to the original LM to create the baseline; since no bigrams are added, backoff weights are not necessary. In addition, the probability mass assigned to the new unigrams is taken directly from <unk>, so renormalization is not required.

Let M be the number of unique OOVs (with respect to the original language model vocabulary) in the original language model training data. Equation (4.1) shows how unigram probabilities are given to each new IV in the baseline.

$$P(newIV) \leftarrow \frac{P_{orig}(\text{<unk>})}{M} \tag{4.1}$$

In our experiments, the actual value for M is 264K. Since the probability mass is taken directly from <unk>, $P(\text{<unk>})$ is also updated as shown in equation (4.2):

$$P(\text{<unk>}) \leftarrow P_{orig}(\text{<unk>}) \times (1 - \frac{N}{M}) \tag{4.2}$$

where N is the number of new IVs added to the language model (in the experiments on the test data, $N = 6382$).

The oracle model is also created as a point of comparison to the adapted models. It represents the upper limit of how much the adapted models can be improved using the contemporary corpus[2]. Like for the baseline model, the vocabulary of the oracle model consists of the vocabulary of the original language model plus the words in the whole-test new IV list. Our oracle model for the test data is created as follows. First, a new bigram language model with Kneser-Ney smoothing is trained on the contemporary corpus described in section 4.1.2. This new language model is interpolated with the original language model, with the interpolation weights trained on a small corpus of transcribed radio speech; the resulting weight of the new model is 0.034.

[2]Further improvements to the models could be made by augmenting the training data (section 4.1.1) or the contemporary corpus (section 4.1.2), or by improving the algorithm for generating the new IV lists (section 4.2). Thus, the oracle model gives us an idea of the optimal performance that can be achieved using the current data and new IV lists.

Table 4.6 shows the number of unigrams and bigrams in the original language model, the baseline, and the oracle model. Note that 6,382 unigrams are added to each of the baseline and oracle LMs. This corresponds to the number of new IVs in the whole-test new IV list. In addition, 179,406 bigrams are added to the oracle model, resulting in an average of 28.1 added bigrams for each new IV.

Model	**Unigrams**	**Bigrams**
original	96,712	40,256,518
baseline	103,094	40,256,518
oracle	103,094	40,435,924

TABLE 4.6: Language model sizes.

4.5 Evaluation

Various methods are applied in order to evaluate the adapted language models. Here, we give an overview of each of these methods and describe the contexts in which they are used in our experiments.

4.5.1 Perplexity

We use perplexity as the main evaluation measure in our experiments. The standard formula for language model perplexity was given in equation (2.7) in section 2.1.3.

We run three sets of language modeling experiments:

- *Development experiments* (section 5.2): experiments on the development data in which a single adapted language model is created for the whole corpus for each adaptation approach
- *Experiment 1* (section 5.3.1): experiments on the test data in which a single adapted language model is created for the whole corpus for each of the best-performing approaches from the development experiments
- *Experiment 2* (section 5.3.2): experiments on the test data in which a separate adapted language model is created for each article for each of the best-performing approaches from the development experiments

In the development experiments and experiment 1, the results give one perplexity value per algorithm, which can be directly compared to the perplexity of the baseline and oracle LMs. However, each algorithm in experiment 2 yields a separate perplexity value for each article. To facilitate comparisons to the baseline in experiment 2, the perplexities of each adapted language model on its corresponding test article are averaged using the following formula:

$$AvgPP = \exp\left(\frac{\sum_i (w_i + s_i) \log PP_i}{\sum_i (w_i + s_i)}\right) \tag{4.3}$$

where w_i, s_i, and PP_i are the number of words in the test article, number of sentences in the test article, and perplexity of the LM on the test article, respectively.

4.5.2 Recognition error rate

For experiment 1 (section 5.3.1), we also run recognition experiments in order to measure the performance of the models within the ASR system. We evaluate our models using both word error rate (WER) and proper noun error rate (PNER). WER was discussed in section 2.1.3; PNER is simply WER but restricted to proper nouns. The reason for using PNER is that all of our new IVs are proper nouns, and we are interested in seeing how the system's performance on these words in particular is affected. In addition, given that we only add proper nouns to our models, there may not be much room for improvement in overall word error rate.

Chapter 5

Experimental Results

In this chapter, we report on each of our experiments and analyze their results. We start by describing the word embedding comparison experiments; this is followed by an explanation of the experiments performed on the development corpus. Then, we give perplexity and recognition results on the test data, with an analysis of the implications of these results. For the test data experiments, we examine two cases: when a single adapted language model is created for the entire test corpus (experiment 1), and when a different model is created for each article (experiment 2).

5.1 Word embeddings comparison experiments

As described in section 3.2, we trained word embeddings using different models (skip-gram, CBOW, and structured skip-gram) and context sizes (two, five, ten, and fifteen words). We evaluated these word embeddings by using them to define the similarity measure in adapted language models created for the development corpus and comparing the resulting perplexities. This allowed us to choose an appropriate word embedding model for the perplexity and recognition experiments performed on the test set (see section 5.3). This section gives the results of these comparisons, as well as a qualitative analysis of the word representations resulting from the different word embedding training methods.

5.1.1 Qualitative comparisons

In table 5.1, we display the most similar IV words to the new IVs 'sochi,' 'instagram,' and 'cuaron' using the different word embedding models and context sizes. This table serves to give an idea of the word similarity measures that result from the different word vector training algorithms; in section 5.1.2, we give actual perplexity results for each of these models.

For comparison, table 5.1 also includes the most similar IVs according to the minimum perplexity algorithm (labeled 'norm minPP') proposed in section 3.5.3. Although this method was not fully implemented and its behavior was not studied in a language modeling application, the qualitative results are interesting to compare. In particular, we see that the minimum perplexity method does relatively well in identifying word class, but tends to be biased towards more common IVs.

From the table, we observe that the context size does not seem to make as much of a difference as the embedding model itself. In particular, structured skip-grams seem more adept at finding the category of the word (city, website, last name); for skip-gram and CBOW, the most similar words do not always fall in the same word class. In particular, for CBOW, common nouns are sometimes given (for example, 'réalisateur' and 'film' for 'cuaron'); there is a clear semantic link between the words, but the syntactic relationship is less evident. In addition, it is interesting to note that this table only shows the most similar IV words to each new IV, but many of the most similar words to these new IVs were in fact OOVs.

Model	**Context**	**sochi**	**instagram**	**cuaron**
skip-gram	15 words	sotchi avrasya kavgolovo	facebook twitter followers	sideways benicio fantastic'arts
skip-gram	10 words	sotchi avrasya kavgolovo	facebook twitter tumblr	adentro benicio sideways
skip-gram	5 words	sotchi kavgolovo trébizonde	facebook twitter tumblr	volver benicio adentro
skip-gram	2 words	sotchi universiade rethymnon	facebook twitter tumblr	adentro volver braveheart
struct. skip	5 words	sotchi varna kusadasi	facebook twitter youtube	banderas soderbergh sinise
CBOW	15 words	sotchi volgograd astrakhan	facebook twitter tumblr	réalisateur bardem film
CBOW	10 words	sotchi dmitry azov	facebook twitter tumblr	réalisateur film adentro
CBOW	5 words	sotchi arkhangelsk volgograd	facebook twitter tumblr	réalisateur film gladiator
CBOW	2 words	sotchi arkhangelsk volgograd	facebook twitter tumblr	réalisateur centelles volver
norm minPP	N/A	barcelone pékin londres	twitter facebook et	cano portillo martinez

TABLE 5.1: Most similar IV words to 'sochi,' 'instagram,' and 'cuaron' using different word similarity measures.

5.1.2 Perplexity results

In order to find the ideal word embeddings for use in the experiments, we created language models using each of the embedding methods for a particular LM adaptation method. We chose a mixed adaptation approach for testing the word embeddings; this algorithm was chosen both because it had a lower perplexity than the baseline (in initial experiments using skip-gram with a context of ten words) and because it used word similarity in various ways. This approach can be defined as follows:

- $P(newIV)$ is maximum probability of five most similar IVs
- $B(newIV)$ is minimum backoff weight of five most similar IVs
- Bigrams are found using the contemporary corpus; bigrams are neither limited (with a cutoff) nor artificially generated
  - $P(x|newIV)$ uniform

  - $P(newIV|x)$ is probability of closest IV with non-zero bigram probability

- For new IVs for which no bigram is added from the contemporary corpus, bigrams based on the closest IV are added
  - Three x-newIV bigrams and three newIV-x bigrams added
  - Bigrams with highest probability added

Table 5.2 shows the perplexity results for the different word embeddings on the development corpus using this adaptation algorithm.

Model	**Context**	**Perplexity**
baseline	N/A	230.39
skip-gram	15 words	225.04
skip-gram	10 words	225.10
skip-gram	5 words	224.63
skip-gram	2 words	**224.42**
struct. skip	5 words	225.16
CBOW	15 words	229.92
CBOW	10 words	231.40
CBOW	5 words	229.67
CBOW	2 words	228.34

TABLE 5.2: LM perplexities resulting from different word embeddings.

5.1.3 Analysis

For the experiments on the test corpus, we chose the skip-gram embeddings with a context of two words, since this was the model that performed the best on the development corpus. However, there did not seem to be much difference among the skip-gram methods or between skip-gram and structured skip-gram. On the other hand, the CBOW embeddings performed significantly worse, in some cases not even improving over the baseline. Our qualitative comparison of the methods in section 5.1.1 revealed that vectors trained with CBOW tended to find similar words to the new IVs that were not in the same word class and were not proper nouns; this could be an explanation for their poor performance on this task.

5.2 Language model adaptation: Development corpus experiments

The language model parameter estimation approaches we proposed in section 3.3 each had multiple variations. In order to decide which algorithms to examine on the test data, we first studied each of the possibilities on the development corpus. In these experiments, we created a single language model for each algorithm for the entire development corpus. We also ran preliminary per-article experiments in order to get an idea of which algorithms might be worth testing in experiment 2 (section 5.3.2). Here, we concentrate on the experiments where a single language model was created for the whole deveopment corpus, but when the per-article LM results differ, we mention them as well.

For efficient experimentation, we divided adaptation up into smaller subtasks (estimation of unigram probabilities, finding bigrams, etc.). We then studied each subtask to find the best approach before moving on to the next task; on subsequent subtasks, we used only the methods from previous subtasks that were found to yield the lowest perplexities. Note that we assumed that each subtask was independent; this strategy may have been sub-optimal, as it is possible that there were interactions between the subtasks.

In the experiments on the test corpus (section 5.3), we took only the algorithms that gave the lowest perplexities on the development corpus. Note that the development experiments used the skip-gram model with a context size of ten words to estimate the word embeddings; however, as noted in section 5.1, we used word embeddings trained with a context size of two words in the experiments on the test data[1].

5.2.1 Unigram probability estimation

Recall that in chapter 3 we proposed various ways of estimating unigram probabilities for new IVs, for both similarity-based and corpus-based approaches. Table 5.3 shows the perplexities on the development corpus when each of these unigram probabilty estimation methods was applied. The first seven adaptation methods are similarity-based approaches; the remaining four are corpus-based approaches.

Unigram estimation method	**Perplexity**
baseline	230.39
oracle	214.02
closest IV probability	229.11
5 similar IVs – maximum probability	**228.20**
5 similar IVs – minimum probability	229.70
5 similar IVs – median probability	229.02
5 similar IVs – average probability	230.86
5 similar IVs – average prob. weighted by word similarity	231.37
5 similar IVs – avg. prob. weighted by % of max. word sim.	230.95
maximum likelihood	227.99
weighted by corpus occurrences	**227.24**
scaled closest IV probability	232.33
scaled maximum probability of similar IVs	231.01

TABLE 5.3: Perplexity results for unigram probability estimates.

We can see from the table that weighting the baseline probability by the number of corpus occurrences performed best overall, while taking the maximum probability of the similar IVs performed best among the similarity-based approaches. As a result, in the following experiments, we used these methods to estimate unigram probabilities. However, we should note that there did not seem to be a large difference between the methods, particularly between using the maximum likelihood corpus probability and weighting the baseline probability by corpus counts.

5.2.2 Bigram probability estimation

Once we found the best similarity-based and corpus-based techniques for estimating unigram probabilities, we studied adding bigrams using each of these approaches. For similarity-based bigram estimation, the most successful similarity-based unigram probability estimation method (maximum probability of five most similar IVs) was used. Likewise, for corpus-based bigram estimation, the unigram probability estimation method that was used was weighting baseline

[1]The word embedding experiments and the language modeling experiments on the development corpus were run concurrently. Therefore, we chose a word embedding model for the development corpus experiments before finding out which algorithm to use on the test corpus experiments.

probabilities by number of corpus occurrences, as that method yielded the lowest perplexity in the unigram estimation experiments.

Similarity-based estimation

The techniques studied in this section were the different similarity-based ways of finding bigrams specified in section 3.3.1. We experimented with the following axes:

- Using bigrams of closest IV vs. bigrams of IV used to find unigram probability
- Limiting the number of bigrams added per new IV to six or 24 bigrams
- Different ways of choosing the bigrams to add

Once we found the method with the lowest perplexity for each axis, we moved onto the next axis, using the best method from all of the previous axes. Note that we chose six and 24 bigrams to add per new IV as heuristics; this is because there were an average of six unique bigrams for each whole-development new IV in the contemporary corpus, while the oracle model on the development data added an average of 24 bigrams per new IV.

We did not experiment with different ways of defining bigram probabilities and unigram backoff. As justified in section 3.3.1, we simply modeled these values after those of the IV word used for finding bigrams.

Table 5.4 shows the perplexity results for these algorithms on the development corpus. Although the lowest perplexity came when adding the 24 bigrams of the closest IV that had the

Bigram estimation method	**Perplexity**
baseline	230.39
oracle	214.02
unigrams only (no new IV bigrams)	228.20
bigrams of closest IV	**230.95**
bigrams of IV used to assign unigram probability	232.27
6 bigrams of the closest IV with highest probability	228.25
24 bigrams of the closest IV with highest probability	**228.20**
24 bigrams of the closest IV with highest probability	**228.20**
24 bigrams of the closest IV with highest % of probability	228.23
24 bigrams of the closest IV with highest prob. relative to max.	233.98
12 bigrams of closest IV of each type with highest prob.	228.24

TABLE 5.4: Perplexity results for similarity-based bigram estimation.

highest probability, it is interesting to note that this method did not improve over the version where no bigrams were added (see table 5.3). In addition, the preliminary per-article results differed from these results in that the best-performing algorithm used all of the bigrams of the IV used (so that the entire behavior of the new IV was modeled after that of the IV used). As a result, we experimented with two methods in experiment 1 on the test data (adding no bigrams and adding the 24 highest-probability bigrams of the closest IV), and in experiment 2 on the test data, we studied a third method (adding all bigrams of the IV used).

Corpus-based estimation

This section contains experiments run on the development data to find the best corpus-based bigram estimates. Like for the similarity-based approaches, we split this task into several axes and found the method that performed the best on each axis before testing the next axis. We studied the following aspects:

- Adding bigrams of the form newIV-newIV
- Limiting bigrams approach with various cutoffs
- Generating bigrams approach
- Backoff weight estimation
- Estimation of $P(x|newIV)$
- Estimation of $P(newIV|x)$

We considered a default for each aspect; this default was used on experiments before the best-performing method was established. The defaults were as follows:

- Include bigrams of the form newIV-newIV
- Do not limit bigrams or generate bigrams
- Backoff weight is $B($<unk>$)$
- $P(x|newIV)$ is uniform
- $P(newIV|x)$ is minimum probability of bigrams starting with x

Table 5.5 displays the perplexity results on the whole development corpus when each of these axes were varied one at a time. Note that when using the generating bigrams approach,

Bigram estimation method	**Perplexity**
baseline	230.39
oracle	214.02
unigrams only (no new IV bigrams)	227.24
no bigrams newIV-newIV	228.23
bigrams newIV-newIV	**227.41**
all corpus bigrams	227.41
limiting bigrams – cutoff of 2	227.36
limiting bigrams – cutoff of 5	**227.25**
limiting bigrams – cutoff of 10	227.28
generating bigrams – similarity cutoff of 0.8	227.38
<unk> backoff	227.25
closest IV backoff	**226.78**
maximum backoff of 5 closest IVs	226.84
minimum backoff of 5 closest IVs	226.79
$P(x\|newIV)$ uniform	**226.78**
$P(x\|newIV)$ weighted by corpus occurrences	230.12
$P(newIV\|x)$ is $\min_y P(y\|x)$	226.78
$P(newIV\|x)$ is $\max_y P(y\|x)$	307.53
$P(newIV\|x)$ is $P(y\|x)$ for closest IV y st. $P(y\|x) > 0$	225.31
$P(newIV\|x)$ is $\max_y P(y\|x)$ for 5 closest IVs y st. $P(y\|x) > 0$	**223.78**

TABLE 5.5: Perplexity results for corpus-based bigram estimation.

we still used a cutoff of five occurrences to include bigrams in the model. There were two reasons for this. First, in the previous experiments, using the cutoff was more successful than including all the bigrams. In addition, it is computationally very expensive to include all bigrams for this algorithm, which goes against our goal of dynamic, on-the-fly adaptation of the models.

For the bigram probabilities of the form $P(x|newIV)$, we did not experiment with weighting them by cosine similarity. Doing so would not make sense, since we did not use the generating bigrams approach (as it had a higher perplexity than the limiting bigrams approach). For the preliminary experiments with a different language model for each article, the only difference in the results was that the performance was best when bigrams from the corpus were given a cutoff of two (instead of five).

From the table, we can see that our best proposed method gave a large reduction in perplexity over the baseline; the reduction was 40.4% of the highest possible reduction (defined using the perplexity of the oracle model). In addition, in contrast to the similarity-based methods described above, the corpus-based bigram methods resulted in an improvement over just adding unigrams.

It is interesting to note that the largest improvements to perplexity came from two aspects: the unigram probability estimations and the estimations of $P(newIV|x)$. By contrast, the methods for finding bigrams and the backoff weight estimations that we proposed did not perform very differently from each other. Given the importance of n-gram probability estimates, it is feasible that further improving the estimation of $P(x|newIV)$ would also cause a reduction in perplexity. Another interesting aspect of the results is that simply adding bigrams from the corpus did not give us an improvement over the unigram case; appropriate estimations of probabilities and backoff weights were needed in order for the corpus bigrams to be usable. In particular, the fact that the perplexity of the model that estimated $P(newIV|x)$ using the maximum probability of all bigrams starting with x was so high shows that it is necessary to take care not to assign too much probability to the new bigrams.

Mixed bigram estimation

We also studied the performance of the mixed approach on the development data. We combined the bigram selection method of the similarity-based approach that gave the lowest perplexity with the best-performing algorithm of the corpus-based approach. For unigram estimation, we used the best corpus-based method, as it outperformed the similarity-based methods.

Table 5.6 displays the perplexity results for the mixed approach on the development corpus. For comparison, in addition to the baseline and oracle language model results, we show the perplexities for the most successful similarity-based and corpus-based algorithms.

Bigram estimation method	**Perplexity**
baseline	230.39
oracle	214.02
similarity-based	228.20
corpus-based	**223.78**
mixed	223.84

TABLE 5.6: Perplexity results for mixed bigram estimation.

We can see from the table that the mixed approach did not outperform the corpus-based approach. In fact, the perplexities were very similar for the two methods. This could be due to the fact that adding bigrams did not cause a decrease in perplexity for the similarity-based approaches. As a result, we did not experiment with this method for the test data, since adding the bigrams based on similarity seemed to simply add slightly more noise to the model. It is possible that a different mixed approach would have yielded improvements; this is something to be explored in the future.

5.2.3 Analysis

The purpose of the experiments reported in this section was to find which methods gave the lowest perplexity on the development data in order to decide which methods to apply to the test data. As a result of these experiments, we chose different methods for the test data experiments in which a single language model is created for the whole corpus (experiment 1; section 5.3.1) and for the per-article experiments on the test data (experiment 2; section 5.3.2). For both cases, we tested both similarity-based and corpus-based methods. Recall also that the experiments on the test data used the skip-gram word embeddings trained on a context of two words; this was determined by the experiments described in section 5.1. We did not use the mixed approach to adding bigrams described in section 5.2.2, since it did not outperform the best corpus-based approach.

For experiment 1 on the test data, we chose the following algorithms:

- Similarity-based, unigrams only: find the five most similar IVs to a given new IV; give the new IV the maximum of the probabilities of these IVs
- Similarity-based, unigrams and bigrams: assign unigram probability as above; assign backoff of the closest IV; add the 24 highest-probability bigrams of the closest IV, replacing the closest IV with the new IV
- Corpus-based: unigram probability is baseline probability weighted by number of corpus occurrences; unigram backoff is backoff of closest IV; bigrams are bigrams from the corpus containing the new IV that appear at least five times; $P(x|newIV)$ uniform; for $P(newIV|x)$, find the five closest IVs y such that $P(y|x)$ is nonzero and take the maximum such probability

Note that we decided to study both adding only unigrams and adding unigrams and bigrams for the similarity-based approaches. This is because adding only unigrams resulted in the same perplexity on the development data as for adding unigrams and bigrams for these methods.

For experiment 2 on the test data, we chose the following algorithms:

- Similarity-based: find the five closest IVs to a given new IV and give the new IV the maximum of their unigram probabilities; model backoff weight and bigram behavior exactly after those of the IV used to assign unigram probability
- Corpus-based: as in experiment 1, but with a cutoff of two occurrences instead of five for bigrams in the corpus

Although the similarity-based approaches performed significantly worse on the development data than the corpus-based approaches, we still tested both configurations in the experiments on the test data. This was done to see whether improvements in recognition could be made without using the contemporary corpus.

Finally, we would like to call attention to the improvements made over the baseline. If we consider the oracle LM perplexity as the lower limit of achievable perplexity using this training data, the best similarity-based approach achieved 13.4% of possible improvement over the baseline. On the other hand, the best corpus-based approach achieved 40.4% of the possible improvement, which is an encouraging result for our methods. In fact, we expect the methods to perform better when using the skip-gram word vectors with a context of two words (instead of a context of ten words, which was used in these experiments) and when a separate adapted LM for each article is created.

5.3 Language model adaptation: Test corpus experiments

5.3.1 Experiment 1: One language model for the entire corpus

In our first experiment on the test corpus, we created a single language model for the entire corpus for each algorithm. Recall from section 5.2.3 that we selected the three best performing algorithms to study on the test corpus. For experiment 1, we examined both language model perplexity and recognition error on the test data.

We created language models for the test corpus using two similarity-based algorithms – one for which only unigrams were added and one for which unigrams and bigrams were added – and one corpus-based algorithm. See section 5.2.3 for precise specifications of these algorithms. These models were compared to the baseline and oracle models in the case of the perplexity experiments and to the original, baseline, and oracle models in the case of the recognition experiments. All adapted language models were created using word embeddings trained with the skip-gram model and a context of two words.

Perplexity results

Table 5.7 displays the perplexities of the baseline, oracle, and adapted language models on the test data. The oracle model is considered the best possible perplexity that could be achieved on this corpus with our adaptation data.

Adaptation method	Perplexity
baseline	212.03
oracle	197.42
similarity-based unigrams	210.19
similarity-based unigrams and bigrams	210.20
corpus-based	206.59

TABLE 5.7: Perplexity results on the test data for experiment 1.

These results are very similar to what was observed for the development data (see section 5.2.2). All of the methods improved over the baseline; however, adding unigrams and bigrams using a similarity-based approach did not do as well as just adding unigrams. In addition, the corpus-based method performed better than either similarity-based method.

In fact, the similarity-based approaches yielded only relatively small improvements. If we take the oracle perplexity as the lower limit for perplexity, then the similarity-based unigram-only method only represented 12.6% of the possible improvement over the baseline. By comparison, the decrease in perplexity using the corpus-based method represented 37.2% of the possible improvement.

Recognition results

The recognition experiments were run using our ASR system, which was based on the Kaldi toolkit[2]. The acoustic model was a hidden Markov model-deep neural network combination with context-dependent phone units. The lexicon of the system contained about 100K words. Our adapted bigram language models were used in the first pass of the system.

The results for the recognition experiments on the whole corpus are shown in table 5.8. We display the word error rate and proper noun error rate on the test data for each of the adapted

[2]http://kaldi-asr.org/

models, as well as for the original, baseline, and oracle language models. Statistical significance was calculated using the matched-pairs significance test described by Gillick and Cox (1989).

Adaptation method	WER	PNER
original LM	33.8	56.8
baseline	33.4	52.3
oracle	33.1	49.9
similarity-based unigrams	33.4	52.1
similarity-based unigrams and bigrams	33.4	52.2
corpus-based	33.2*	50.9*

TABLE 5.8: Recognition error rates on the test data for experiment 1. Asterisks indicate statistically significant improvements over the baseline for adapted models.

An interesting result is that our baseline model (in which a very small amount of probability was given to all new IVs) improved significantly over the original LM in both WER and PNER. This indicates that even naively assigning probability mass to new IVs is helpful in recognition. Although the differences between the similarity-based models and the baseline were not statistically significant, the corpus-based model did yield a significant improvement over the baseline in both WER and PNER.

Language model statistics

In order to investigate the factors affecting the performance of the adapted language models, we further analyzed the unigrams and bigrams added to the models. A total of 6,382 unigrams (corresponding to the 6,382 new IVs) were added to each of the language models; the resulting LMs had 103,094 unigrams. However, of the new IVs, only 202 actually occurred in the test data; furthermore, 117 of them occurred only once. In fact, there were only 1,188 OOV types in the test data (with respect to the original language model vocabulary). This underscores the difficulty of adding such new IVs to the language model; we are not very confident that these words will appear in the data, so we have to be careful not to take too much probability from words in the original vocabulary to give to the new IVs.

In addition to unigrams, bigrams were also added to the oracle language model, as well as to two of the adapted language models. No bigrams were added to create the similarity-based unigram-only adapted model or the baseline model. In table 5.9, the number of bigrams added to the models is displayed. It is interesting to note that the two best-performing adapted models, the corpus-based model and the similarity-based unigram-only model, added far fewer bigrams than the oracle model. On the other hand, the similarity-based bigram model added a similar number of bigrams to the oracle. However, the fact that the corpus-based model outperformed the similarity-based bigram model indicates that the bigrams added by the corpus-based model may have been more relevant to the test data, despite the fact that less than one bigram was added for every two new IVs.

We also wanted to examine how well the new IV bigrams in the test data were modeled. A total of 648 unique bigrams (bigram types) and 878 total bigrams (bigram tokens) containing new IVs occurred in the test data. Table 5.10 displays what percentages of these new IV bigram types and tokens were added to the language models. Note that the table does not include those models for which no bigrams were added, as the new IV bigram coverage would be zero in those cases.

As we can see from the table, the corpus-based model gave much better coverage of the bigrams containing new IVs than the similarity-based model, despite adding far fewer new IV

Adaptation method	Bigrams added
baseline	0
oracle	179,406
similarity-based unigrams	0
similarity-based unigrams and bigrams	137,544
corpus-based	2,692

TABLE 5.9: Number of bigrams added to each language model.

Adaptation method	Bigram types (%)	Bigram tokens (%)
oracle	46.6	57.7
similairty-based unigrams and bigrams	22.8	20.5
corpus-based	23.3	36.6

TABLE 5.10: Coverage of bigrams in the test data containing new IVs for each language model.

bigrams. In addition, the bigrams that were added to the corpus-based model tended to be much more relevant, as evidenced by the fact that roughly the same number of bigram types covered far more bigram tokens in the corpus-based approach than in the similarity-based approach.

Analysis

Our corpus-based approach for language model estimation improved over the baseline LM in both perplexity and recognition; the improvement in perplexity was 37.2% of possible improvement (based on the oracle LM). This approach also showed a statistically significant reduction in both WER and PNER over the original LM and the baseline. These results indicate that our proposed corpus-based approaches can be successful for adding words to the language model of an ASR system.

On the other hand, the similarity-based approaches showed only very small improvements over the baseline in perplexity, and were not significantly different from the baseline in either PNER or WER. In addition, the similarity-based algorithm in which bigrams were added did not yield better results than the similarity-based unigram-only algorithm.

These results were quite similar to the results on the development data in section 5.2. The fact that the similarity-based unigram-only method outperformed the similarity-based algorithm in which bigrams were added indicates that our similarity-based method for adding bigrams was not successful; this is a point that should be explored in future work. This hypothesis is further supported by the bigram coverage of each model (see table 5.10); despite adding more bigrams to the LM, the similarity-based method yielded a lower coverage of the new IV bigrams in the test data.

The results from the recognition experiments mirrored those of the perplexity experiments very closely. Algorithms that yielded a lower perplexity were more likely to give lower PNER and WER in the recognition experiments. This is important to note because the per-article algorithms in experiment 2 (section 5.3.2) were evaluated using perplexity only; our results here suggest that perplexity is a good predictor of performance in recognition experiments. In addition, our corpus-based model showed improvements in both PNER and WER despite adding only proper nouns to the model; thus, adding proper nouns seems to help the system overall.

5.3.2 Experiment 2: Different language model for each article

Our second experiment on the test corpus was a closer match to the original goals of the project. In this experiment, we used a separate new IV list for each article in the test data to create a separate language model for each article for each algorithm. The tailored new IV lists allowed the resulting LMs to better model each specific article. For these experiments, we only calculated the perplexity of the models on each article. Recognition experiments were not possible in this experimental setup because of the time required to compile LMs in our current ASR system; however, in the updated system (currently in progress), per-article recognition experiments will be feasible.

We used the similarity-based and corpus-based approaches that yielded the lowest perplexities on the development data to create adapted language models for the test data in experiment 2. See section 5.2.3 for a full specification of the two algorithms used. As in experiment 1, we used skip-gram word embeddings trained with a context of two words to calculate word similarity, and we compared the average perplexity of the models on the test articles with the perplexities of the oracle and baseline LMs.

Perplexity results

In table 5.11, the average perplexities for each of the algorithms on the test data are displayed. Note that in the case of the baseline and the oracle LMs, the average perplexity on the test corpus is the same as the perplexity on the whole corpus (in table 5.7). This is because baseline and oracle models were not created on a per-article basis; only one of each model was created for the whole test corpus.

Adaptation method	Avg. PP
baseline	212.03
oracle	197.42
similarity-based	208.91
corpus-based	205.93

TABLE 5.11: Average test data perplexity for experiment 2.

We observe similar patterns to what we saw for the whole corpus experiments (see section 5.3.1). Both of our models improved over the baseline, and the corpus-based model showed about twice as much improvement as the similarity-based model. In total, taking the oracle perplexity as the lower limit, the similarity-based model yielded 21.4% and the corpus-based model 41.8% of the possible improvement over the baseline. Note that, as expected, our algorithms performed better in the per-article experiments than in the whole corpus experiments (experiment 1).

Article statistics

It is interesting to note that, although both models outperformed the baseline overall, the baseline gave the lowest perplexity on 283 of the 467 articles (60.6%). The reason for the better overall performance of the adapted models was twofold. First, in the articles for which the adapted models did not improve over the baseline, they did not significantly increase the perplexity, either – the maximum perplexity percent increase on an article was 1.93% for the similarity-based model and 0.50% for the corpus-based model. Similarly, the decreases in perplexity were much higher compared to the baseline. The average perplexity percent decreases for the similarity-based model and the corpus-based model were 3.81% and 7.66%, respectively. This means that these methods were successful in not increasing the perplexity too much when adding words to the LM that do not actually occur in the article, while also modeling the new IVs well when they did appear in the relevant articles.

The number of OOVs (with respect to the vocabulary of the original LM) in an article made a difference in how well the models performed on that article. The articles for which the adapted models performed worse contained fewer OOVs on average than those for which the models performed better than the baseline (2.9 vs. 4.6 OOV types for the word similarity method, 3.0 vs. 4.6 OOV types for the corpus-based method). This makes sense, as we can think of the number of OOVs in an article as the room for improvement by the adapted models over the baseline.

Analysis

The results of the per-article experiments showed that our models can successfully add new words to a language model on a dynamic basis. In particular, the corpus-based method performed almost twice as well as the similarity-based method, although both methods yielded improvements over the baseline. These results mirror what we found in the experiments on the development data (section 5.2) and in experiment 1 on the test data (section 5.3.1). One caveat to these results was that we did not perform recognition experiments on these models; hence, we cannot be completely certain that our models would improve the performance of the overall system. However, given that the per-article LMs gave a lower perplexity on the test corpus than the LMs created in experiment 1 (section 5.3.1), it is not unreasonable to expect some improvement in WER and PNER over the models from experiment 1.

Further per-article analysis of our results showed that our approaches did not outperform the baseline for most of the articles; in particular, articles with fewer OOVs were less likely to be improved by the adapted models. However, the adapted models succeeded in not significantly worsening the performance of the language models on articles with fewer OOVs (where the added new IVs were more likely to be noise). Additionally, our approaches lowered perplexity quite a bit in articles with more OOVs, resulting in an improvement in average perplexity.

5.4 Discussion

This chapter included results on word embedding comparisons, development data experiments, and test data experiments. In this section, we give a summary and an analysis of the experiments presented in this chapter. We also offer possible explanations for the results that we have seen.

We compared the performances of word embeddings trained using different models and context sizes in our language model estimation methods and found that skip-gram vectors with a context of two words yielded the lowest perplexity. This indicated the importance of encoding syntactic as well as semantic information within our vectors. However, qualitative analysis indicated that all of the word embeddings were relatively similar to each other; using a radically different word similarity measure may yield different results.

In the development experiments, we found that the corpus-based approaches yielded lower perplexities than the similarity-based approaches. In addition, for estimating unigram probabilities, methods that assigned a relatively high probability to the new IVs tended to perform better than other methods. This indicates that giving high unigram probability to new IVs is a successful tactic; in particular, using more probable similar words (like the ones that result from the minimum perplexity method described in section 3.5.3) might make sense. Results on experiments with adding bigrams, however, indicated that giving a high probability to new IV bigrams may not always be an improvement (see, for example, line 18 in table 5.5). In fact, in the corpus-based methods, estimating bigram probabilities using similar IVs yielded the greatest improvement compared to other methods and subtasks.

Experiment 1 (one language model for the whole corpus) and experiment 2 (one language model per article) yielded similar results. In both cases, the similarity-based approaches resulted in small improvements over the baseline, while the corpus-based approaches performed even better. In addition, the improvements in experiment 2 were more pronounced for all approaches than in experiment 1. No recognition experiments were run for experiment 2; however, the similarity between the perplexity and recognition results for experiment 1 and the fact that experiment 2 gave better perplexity results than experiment 1 indicate that using the per-article language models of experiment 2 might result in an even greater improvement in PNER and WER than for experiment 1.

Further analysis of the results showed that our models were effective in assigning probability to n-grams containing new IVs that actually occurred in the articles, without damaging the models by assigning too much probability to n-grams containing new IVs that did not occur. One factor that may have contributed to this was the use of the similar IVs to the new IV; as mentioned before, these IVs are often words that are rare themselves, and thus that have a lower probability. In the case of the corpus-based models, the use of the cutoff of corpus occurrences in order to count bigrams was probably helpful in limiting the bigrams added to the model to only those that were most likely to occur. Inflating unigram probabilities by weighting them based on corpus occurrences seems to have helped to choose the most probable new IVs as well, while avoiding giving too much probability to the new IVs that did not occur. These factors depended on the contexts corpus being similar to the test data, at least in the distribution of proper nouns.

The corpus-based methods outperforming the similarity-based methods is somewhat surprising, since similarity-based methods seem to be more common in the literature. This might indicate that, given a small amount of data, it would make more sense to use that data for direct estimation of n-grams rather than to use it to define word similarity (although it should be noted that our corpus-based methods also took advantage of word similarity to estimate probabilities).

One explanation for why corpus-based methods yielded lower perplexities than similarity-based methods could simply be that more data (i.e., the contemporary corpus) was used in the corpus-based methods. Another explanation could be that word embeddings are not ideal for bigram language model estimation, since they tend to encode longer-span, semantic information; a different similarity measure might have yielded a better performance. Finally, in experiment 1 on the test data, the similarity-based approach was more successful when no bigrams were added. Further analysis (see table 5.10) indicated that although many more bigrams were added for each new IV using similarity-based approaches than using corpus-based approaches, the corpus-based approaches gave a better coverage of the new IV bigrams, particularly at the token level. This is an indication that the corpus-based approaches were more able to select bigrams that were likely to occur (and occur multiple times) than the similarity-based approaches. Hence, there is room for improvement in the similarity-based methods for selecting bigrams.

Chapter 6

Conclusions and Future Work

6.1 Discussion

In this work, we set out to explore ways of improving recognition of OOVs in an ASR system by adding OOVs to the bigram language model as new IVs. This was done by dynamically updating the language model without retraining it. We proposed two strategies: a similarity-based approach that took advantage of word embeddings to model behavior of new IVs after that of IVs close to them in the vector space, and a corpus-based approach that used a small corpus from the same time period as the test data to find bigrams and estimate probabilities. The models we proposed yielded improvements in perplexity over the baseline; in addition, the corpus-based approach led to a decrease in WER and PNER over the baseline in recognition experiments.

Experiment 2, in which a different adapted language model was created for each article in the test data, illustrated the advantages of our methods. For articles that did not contain any OOVs (with respect to the original language model), there was no room for improvement, as our new IV list and adaptations just added noise to the LM. However, our methods gave perplexities that were very similar to the baseline perplexities for these cases. On the other hand, for articles with many OOVs, our models tended to show large improvements over the baseline; the corpus-based approaches did particularly well. This indicates that our proposed models were able to assign a high probability mass to the new IVs and their bigrams when they occurred in the test data, while avoiding taking too much probability mass away from existing words in the LM.

Our corpus-based approach yielded a statistically significant decrease in WER and PNER over the baseline in experiment 1. However, there was no significant difference between the similarity-based approaches and the baseline in PNER or WER. Note that we observed that perplexity results and recognition results were very similar in experiment 1; in addition, the per-article adapted LMs of experiment 2 outperformed the whole corpus adapted LMs of experiment 1 in perplexity. This suggests that recognition experiments using the experimental setup of experiment 2 (creating one adapted LM per article) could show to an even greater decrease in PNER or WER.

6.2 Future work

The overall success of our models indicates that continued research in this direction could be fruitful. In this section, we describe some extensions to our experiments that can give more information about the structure and behavior of our models, as well as some suggested directions for future work in the long term.

6.2.1 Short-term improvements

One important step for further examining the performance of our models is to run recognition experiments per article, instead of just for the whole corpus; this would allow us to see if the improvements in perplexity would carry over to improvements in recognition error rate. In addition, we would like to experiment with different sizes for the contemporary corpus. This would give us insight into the corpus size at which the corpus-based models are outperformed by the similarity-based models and the effect of corpus size on unigram probabilities (which were estimated using raw corpus counts).

The methods we used to find the best-performing LM adaptation algorithm had some limitations that could be improved in future work. First of all, as described in section 5.2, we broke our estimation task into subtasks and made the assumption that these subtasks were independent. In the future, we plan on further exploring the interaction between these subtasks. Similarly, some of our hyper-parameters (such as the number of most similar IV words to use or the number of bigrams to add to the LM) were decided using intuition. We will improve our method for choosing these parameters, such as training them on development data.

Finally, our results are difficult to compare to previous work done in this area, since we relied on automatically generated lists of relevant words. The amount that our results depended on these lists is not clear; thus, we hope to re-run our experiments using other methods for finding new IVs in order to compare our results to those achieved by existing algorithms.

6.2.2 Long-term improvements

There are many directions to be explored in the long term based on this thesis. In section 3.5, we gave specifications of methods for adapting trigram LMs, improving renormalization, and using a language model-based word similarity measure. In future work, we will prioritize implementing these methods, as they directly address weaknesses of our existing models.

Improvements could also be made with regards to the word similarity methods. Since the different methods seemed to encode different information, we could combine them in some way, such as by using different methods for different subtasks; another option would be to choose words that are similar to a given new IV in multiple models, which might reduce model-based noise. Adding character-level or morphological information to augment word embeddings could also be helpful, particularly for modeling rare words or words with no vector representations at all. In addition, since the new IVs are all proper nouns, finding only the similar IVs that are also proper nouns might improve our models.

There are a number of other areas in which we could tweak our models. For example, we only studied adding proper nouns to the language models; it would be interesting to see if adding other word classes yields similar results. In addition, although probabilities for existing n-grams were changed in our methods, none of the existing backoff weights were modified; it would be good to explore the effects of changing these weights as well.

This thesis focused on adding words to language models by directly estimating parameters based on word similarity measures and a small contemporary corpus. However, different approaches could be explored in the future. These could include class-based modeling, using intelligent data selection strategies to create a better contemporary corpus, or applying language model pruning methods to select the bigrams to add to the language model.

6.3 Closing remarks

The goal of this work was to devise algorithms for dynamically adding relevant out-of-vocabulary words to a language model. We introduced a number of methods for estimating language model parameters based on word similarity (using word embeddings) and a small contemporary corpus. These methods were successful in creating adapted models that improved in perplexity over the baseline; in addition, our corpus-based models improved in recognition error rate (PNER and WER) over the baseline. Importantly, unlike previous work on this topic, our system added a large number of new IV words to relatively short articles; our models were successful in handling this challenge by improving the LM when the new IVs occurred and not damaging it when they did not occur in the article. In addition, we closely examined the resulting models in order to identify areas in which improvements can be made; we recommended directions for such improvements in section 6.2.

The research done in this thesis shows that language models can be temporally adapted using a very small amount of data; the methods explored here could be carried over to similar areas, such as domain adaptation. We also studied the use of word embeddings for augmenting n-gram language models. This allowed the incorporation of longer-span information into a language modeling method that normally has a very short memory. In addition to applications of this research to domain adaptation and language modeling, the practical applications are clear. The dynamic adaptation methods introduced here would be beneficial in a speech recognition system for broadcast news, since they would allow new models containing a more relevant vocabulary to be created for each news article.

Bibliography

Allauzen, Alexandre and Jean-Luc Gauvain (2005). "Open vocabulary ASR for audiovisual document indexation". In: *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE, pp. 1013–1016.

Bahl, Lalit R., Frederick Jelinek, and Robert L. Mercer (1983). "A maximum likelihood approach to continuous speech recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5 (2), pp. 179–190.

Bansal, Mohit, Kevin Gimpel, and Karen Livescu (2014). "Tailoring continuous word representations for dependency parsing". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Vol. 2. ACL, pp. 809–815.

Baroni, Marco, Georgiana Dinu, and Germán Kruszewski (2014). "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Vol. 1. ACL, pp. 238–247.

Bellegarda, Jerome R. (2004). "Statistical language model adaptation: Review and perspectives". In: *Speech Communication* 42 (1), pp. 93–108.

Bengio, Yoshua et al. (2003). "A neural probabilistic language model". In: *Journal of Machine Learning Research* 3, pp. 1137–1155.

Botha, Jan A. and Phil Blunsom (2014). "Compositional morphology for word representations and language modeling". In: *arXiv preprint arXiv:1405.4273*.

Brown, Peter F. et al. (1990). "A statistical approach to machine translation". In: *Computational Linguistics* 16 (2), pp. 79–85.

Bullinaria, John A. and Joseph P. Levy (2007). "Extracting semantic representations from word co-occurrence statistics: A computational study". In: *Behavior Research Methods* 39 (3), pp. 510–526.

Chen, Stanley F., Douglas Beeferman, and Roni Rosenfeld (1998). *Evaluation metrics for language models*. Tech. rep. Carnegie Mellon University.

Chen, Stanley F. and Joshua Goodman (1999). "An empirical study of smoothing techniques for language modeling". In: *Computer Speech & Language* 13 (4), pp. 359–393.

Clarkson, Philip and Tony Robinson (1998). "The applicability of adaptive language modeling for the broadcast news task". In: *Proceedings of the Fifth International Conference on Spoken Language Processing*. ESCA.

Collobert, Ronan et al. (2011). "Natural language processing (almost) from scratch". In: *Journal of Machine Learning Research* 12, pp. 2493–2537.

Dhillon, Paramveer, Dean P. Foster, and Lyle H. Ungar (2011). "Multi-view learning of word embeddings via CCA". In: *Advances in Neural Information Processing Systems 24*. NIPS Foundation, pp. 199–207.

Gillick, Laurence and Stephen J. Cox (1989). "Some statistical issues in the comparison of speech recognition algorithms". In: *Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing*. IEEE, pp. 532–535.

Goodman, Joshua T. (2001). "A bit of progress in language modeling". In: *Computer Speech & Language* 15 (4), pp. 403–434.

Harris, Zellig S. (1954). "Distributional structure". In: *Word* 10 (2–3), pp. 146–162.

Illina, Irina, Dominique Fohr, and Denis Jouvet (2011). "Grapheme-to-phoneme conversion using conditional random fields". In: *Proceedings of the Twelfth Annual Conference of the International Speech Communication Association*. ISCA, pp. 2313–2316.

Ito, Akinori, Masaki Kohda, and Mari Ostendorf (1999). "A new metric for stochastic language model evaluation". In: *Proceedings of the Sixth European Conference on Speech Communication and Technology*. ESCA, pp. 1591–1594.

Iyer, Rukmini, Mari Ostendorf, and Marie Meteer (1997). "Analyzing and predicting language model improvements". In: *Proceedings of the 1997 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, pp. 254–261.

Johnson, Sue E. et al. (1999). "The Cambridge University spoken document retrieval system". In: *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE, pp. 49–52.

Klakow, Dietrich (1998). "Log-linear interpolation of language models". In: *Proceedings of the Fifth International Conference on Spoken Language Processing*. ESCA.

Klakow, Dietrich and Jochen Peters (2002). "Testing the correlation of word error rate and perplexity". In: *Speech Communication* 38 (1–2), pp. 19–28.

Kuhn, Roland and Renato De Mori (1990). "A cache-based natural language model for speech recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (6), pp. 570–583.

Kukich, Karen (1992). "Techniques for automatically correcting words in text". In: *ACM Computing Surveys* 24 (4), pp. 377–439.

Kullback, Solomon and Richard A. Leibler (1951). "On information and sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86.

Lecorvé, Gwénolé, Guillaume Gravier, and Pascale Sébillot (2011). "Automatically finding semantically consistent n-grams to add new words in LVCSR systems". In: *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, pp. 4676–4679.

Levy, Omer and Yoav Goldberg (2014a). "Dependency-based word embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Vol. 2. ACL, pp. 302–308.

— (2014b). "Neural word embedding as implicit matrix factorization". In: *Advances in Neural Information Processing Systems 27*. NIPS Foundation, pp. 2177–2185.

Levy, Omer, Yoav Goldberg, and Ido Dagan (2015). "Improving distributional similarity with lessons learned from word embeddings". In: *Transactions of the Association for Computational Linguistics* 3, pp. 211–225.

Ling, Wang et al. (2015). "Two/too simple adaptations of word2vec for syntax problems". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL, pp. 1299–1304.

Liu, Xunying, Mark J. F. Gales, and Philip C. Woodland (2013). "Use of contexts in language model interpolation and adaptation". In: *Computer Speech & Language* 27 (1), pp. 301–321.

Luong, Thang, Richard Socher, and Christopher D. Manning (2013). "Better word representations with recursive neural networks for morphology". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. ACL, pp. 104–113.

Ma, Wei-Yun et al. (2014). *Language model adaptation through shared linear transformations*. Tech. rep. Microsoft Research.

Martins, Ciro, António J.S. Teixeira, and João Paulo Neto (2008). "Automatic estimation of language model parameters for unseen words using morpho-syntactic contextual information". In: *Proceedings of the Ninth Annual Conference of the International Speech Communication Association*. ISCA, pp. 1602–1605.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic regularities in continuous space word representations". In: *Proceedings of the 2013 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies. ACL, pp. 746–751.

Mikolov, Tomas et al. (2013a). “Distributed representations of words and phrases and their compositionality”. In: *Advances in Neural Information Processing Systems 26*. NIPS Foundation, pp. 3111–3119.

Mikolov, Tomas et al. (2013b). “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781*.

Mnih, Andriy and Koray Kavukcuoglu (2013). “Learning word embeddings efficiently with noise-contrastive estimation”. In: *Advances in Neural Information Processing Systems 26*. NIPS Foundation, pp. 2265–2273.

Nanjo, Hiroaki and Tatsuya Kawahara (2005). “A new ASR evaluation measure and minimum Bayes-risk decoding for open-domain speech understanding”. In: *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE, pp. 1053–1056.

Naptali, Welly, Masatoshi Tsuchiya, and Seiichi Nakagawa (2012). “Class-based n-gram language model for new words using out-of-vocabulary to in-vocabulary similarity”. In: *IEICE Transactions on Information and Systems* E95-D.9, pp. 2308–2317.

Ney, Hermann, Ute Essen, and Reinhard Kneser (1995). “On the estimation of ‘small’ probabilities by leaving-one-out”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (12), pp. 1202–1212.

Orosanu, Luiza and Denis Jouvet (2015). “Adding new words into a language model using parameters of known words with similar behavior”. In: *Proceedings of the First International Conference on Natural Language and Speech Processing*.

Park, Youngja et al. (2008). “An empirical analysis of word error rate and keyword error rate”. In: *Proceedings of the Ninth Annual Conference of the International Speech Communication Association*. ISCA, pp. 2070–2073.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). “GloVe: Global vectors for word representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. ACL, pp. 1532–1543.

Ponte, Jay M. and W. Bruce Croft (1998). “A language modeling approach to information retrieval”. In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, pp. 275–281.

Pražák, Aleš, Pavel Ircing, and Luděk Müller (2007). “Language model adaptation using different class-based models”. In: *Proceedings of the 2007 International Conference on Speech and Computer*. Springer-Verlag, pp. 449–454.

Qin, Long (2013). “Learning out-of-vocabulary words in automatic speech recognition”. PhD thesis. Carnegie Mellon University.

Rosenfeld, Ronald (1996). “A maximum entropy approach to adaptive statistical language modeling”. In: *Computer Speech & Language* 10 (3), pp. 187–228.

— (2000). “Two decades of statistical language modeling: Where do we go from here?” In: *Proceedings of the IEEE* 88 (8), pp. 1270–1278.

Rubenstein, Herbert and John B. Goodenough (1965). “Contextual correlates of synonymy”. In: *Communications of the ACM* 8 (10), pp. 627–633.

Rubinstein, Dana et al. (2015). “How well do distributional models capture different types of semantic knowledge?” In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Vol. 2. ACL, pp. 726–730.

Sandness, Eric D. and I. Lee Hetherington (2000). “Keyword-based discriminative training of acoustic models”. In: *Proceedings of the Sixth International Conference on Spoken Language Processing*. ISCA, pp. 135–138.

Shannon, Claude E. (2001). "A mathematical theory of communication". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5 (1), pp. 3–55.

Sheikh, Imran et al. (2015a). "Learning to retrieve out-of-vocabulary words in speech recognition". In: *arXiv preprint arXiv:1511.05389*.

— (2015b). "OOV proper name retrieval using topic and lexical context models". In: *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, pp. 5291–5295.

— (2016a). "Document level semantic context for retrieving OOV proper names". In: *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, pp. 6050–6054.

— (2016b). "Improved neural bag-of-words model to retrieve out-of-vocabulary words in speech recognition".

Sundermeyer, Martin, Hermann Ney, and Ralf Schlüter (2015). "From feedforward to recurrent LSTM neural networks for language modeling". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23 (3), pp. 517–529.

Tong, Xiang and David A. Evans (1996). "A statistical approach to automatic OCR error correction in context". In: *Proceedings of the Fourth Workshop on Very Large Corpora*. ACL, pp. 88–100.

Turney, Peter D. and Patrick Pantel (2010). "From frequency to meaning: Vector space models of semantics". In: *Journal of Artificial Intelligence Research* 37, pp. 141–188.

Ueberla, Joerg (1994). "Analyzing a simple language model—some general conclusions for language models for speech recognition". In: *Computer Speech & Language* 8 (2), pp. 153–176.

Wang, Ye-Yi, Alex Acero, and Ciprian Chelba (2003). "Is word error rate a good indicator for spoken language understanding accuracy". In: *Proceedings of the 2003 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, pp. 577–582.

Yamazaki, Hiroki et al. (2007). "Dynamic language model adaptation using presentation slides for lecture speech recognition". In: *Proceedings of the Eighth Annual Conference of the International Speech Communication Association*. ISCA, pp. 2349–2352.