SAARLAND UNIVERSITY
UNIVERSITY OF LORRAINE

MASTER'S THESIS

Leveraging Word Contexts in Wikipedia for Out-of-Vocabulary Proper Nouns Recovery in Speech Recognition

---

Author: Badr ABDULLAH

Submitted in partial fulfilment of the requirements for the degree Master of Science (MSc) in Language Science and Technology, as part of the European Masters Program in Language and Communication Technologies (LCT) at Saarland University and the University of Lorraine

Supervisors: Prof. Irina ILLINA, University of Lorraine
Prof. Dominique FOHR, CNRS, INRIA
Prof. Dietrich KLAKOW, Saarland University

Based on a research conducted during an internship in

MULTISPEECH Team
INRIA, LORIA, University of Lorraine (France)

December 15, 2018

Eidesstattliche Erklärung

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.

Saarland University, Saarbrücken

Signed:

_____

Date:

_____

"Most of the problems of the world stem from linguistic mistakes and simple misunderstanding. Don't ever take words at face value. When you step into the zone of love, language, as we know it becomes obsolete. That which cannot be put into words can only be grasped through silence."

Rumi

SAARLAND UNIVERSITY
UNIVERSITY OF LORRAINE

Abstract

MULTISPEECH Team
INRIA, LORIA, University of Lorraine (France)

M.SC. in Language and Communication Technologies (LCT)

Leveraging Word Contexts in Wikipedia for Out-of-Vocabulary Proper Nouns Recovery in Speech Recognition

by Badr ABDULLAH

Automatic Speech Recognition (ASR) systems are usually trained on static data and a finite vocabulary. When a spoken utterance contains out-of-vocabulary (OOV) words, ASR systems misrecognize these words as in-vocabulary words with similar acoustic properties, but with entirely different meaning. The majority of OOV words are information-rich proper nouns (e.g., person names, geographic locations, commercial products) that are vital to spoken content understanding. Therefore, failing to recognise OOV words has a significant adverse impact on many downstream applications such as spoken document indexing and translation.

In this thesis, we approach this problem by dynamically extending the ASR vocabulary based on the context obtained from the ASR initial hypothesis. In other words, given the in-vocabulary transcription of a spoken utterance, the goal is to retrieve a ranked list of OOV proper nouns that are likely relevant to the spoken context, add words in this list to the ASR lexicon, and perform a second-pass decoding with the ASR system. To this end, we explore different techniques that leverage topical contexts of OOV words in Wikipedia to develop neural models for OOV words retrieval. Our experiments show that document-specific extension of the ASR vocabulary improves the performance of the speech recognizer.

Acknowledgements

First and foremost, I would like to thank my supervisors, Irina Illina, Dominique Fohr, and Dietrich Klakow, for their invaluable support and guidance throughout this research project. I would also like to thank all the lovely people I met at Multispeech team, particularly my office mates during the research internship; Anastasiia, Ajinkya, Théo, and Guillaume.

I would like to thank Christophe Cerisara from LORIA for his motivating and insightful feedback on the work presented in this thesis. I have learned a lot from Christophe and his in-depth knowledge of modern deep learning approaches as well as conventional speech recognition. My discussions with him were always fruitful.

I would also like to express my gratitude to the European Union for giving me the opportunity to study as an Erasmus Mundus scholarship holder with the LCT program. Being a member of the LCT community has been a fantastic life-changing experience. Special thanks go to Bobbye Pernice for her kindness and support starting from the first day I arrived in Europe until the day I submitted this thesis. I would additionally like to extend a heartfelt thank you to all the professors, colleagues, and fellow students whom I encountered during this course of study.

Finally, I would like to thank my family and friends for their unconditional love and support, without which this thesis would not have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Automatic speech recognition, hereafter referred to as ASR, is a field at the intersection of signal processing, natural language processing (NLP), and pattern recognition. The goal of ASR systems is to convert an acoustic signal of a spoken utterance into its underlying textual representation (i.e., word or character sequence.) The development of ASR systems has been the subject of intensive research and engineering for decades. Therefore, spoken language recognition and understanding research have made tremendous advances and the technology is now deployed in many applications that involve human-computer interaction (HCI) such as voice-controlled devices and spoken dialogue systems.

Modern ASR systems are developed using well-defined statistical frameworks that acquire knowledge about the language from actual realisations of speech data and their corresponding human-generated textual transcriptions. Even though these statistical frameworks work very well in practice, they make a simplifying assumption about the lexicon of the spoken language. That is, statistical ASR methods assume that speakers are going to use words within a finite word vocabulary. This assumption is unrealistic since new words —such as proper nouns, loan words, or newly coined terms— are continuously introduced in the media. Therefore, ASR systems cannot correctly recognise words beyond their finite, predefined lexicon.

The goal of this master's thesis is to develop methods for dynamically extending the lexicon of an ASR system to accommodate new words. The methods proposed in this thesis were inspired by the recent advances in NLP research which rely on distributed word representations, or word embeddings, and deep neural networks. We propose a solution for document-specific extension of the ASR vocabulary in order to correctly transcribe words that do not exist in the base vocabulary of the ASR system.

1.2 The OOV Problem

The so-called Large-Vocabulary Continuous Speech Recognition Systems (LVCSR) systems are trained to recognise words within a large, but finite, word vocabulary (usually referred to the as ASR lexicon). When a spoken utterance contains words that do not exist in the ASR lexicon, these words are going to be incorrectly recognised as in-vocabulary words

with similar acoustic properties, but with entirely different meaning. In the speech recognition community, this limitation of speech recognisers is commonly known as the out-of-vocabulary (OOV) words problem. Figure 1.1 shows an illustrated example of this problem. In an English ASR system, if the two words that constitute the foreign named entity *Slobodan Milosevic* are OOV words, a similar sounding string of words will be hypothesised instead.

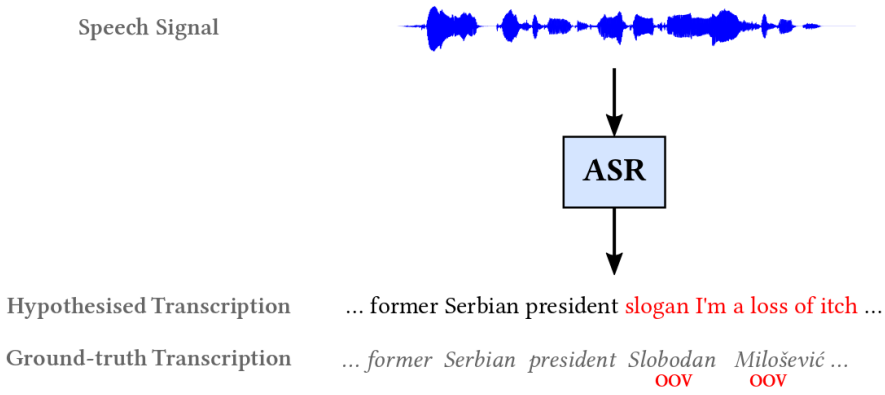

FIGURE 1.1: An example of the OOV words problem.

The majority of OOV words are information-rich proper nouns such as person names, geographic locations, and commercial products. Failing to recognise OOV words hinders the usability of ASR systems as a prerequisite step in the natural language understanding (NLU) pipeline. One the one hand, misrecognizing OOV named entities can cause severe damage on transcript coherence since these entities are vital to spoken content understanding. On the other hand, downstream applications such as indexing, retrieval, and translation of information in spoken documents rely on high-quality automatic transcriptions. Recognition errors caused by OOV proper nouns in automatic transcriptions propagate through these downstream applications. For these reasons, ASR systems that can correctly transcribe content-relevant proper nouns are of great interest.

One might think that an extremely large predefined lexicon with all words of the spoken language would solve the OOV words problem. There are theoretical and practical reasons why this idea does not work for speech recognition. From a theoretical point of view, it not possible to restrict the language usage to be confined within a finite lexicon. For example, a language may borrow words from other languages to denote concepts that have not yet been lexicalised. Moreover, the spoken broadcast news usually cover events that take place in different countries, which introduce new words of multilingual nature on a daily basis. Thus, no finite lexicon is large enough to cover all words that might be encountered in speech. From a practical point of view, an arbitrary extension of the lexicon makes the ASR decoding search space more complex hence introducing more confusability. Therefore, a larger lexicon usually degrades the performance of ASR systems and does not lead to an optimal solution to the OOV words problem.

1.3 Thesis Statement

As explained above, it is important for an ASR system to be able to correctly recognise content-relevant proper nouns, even if the ASR base vocabulary does not include those proper nouns. In this thesis, we propose an approach to recover OOV words that have been incorrectly transcribed by the ASR in the initial hypothesis. Precisely, we explore techniques of dynamically adapting the ASR lexicon based on the semantic context of each spoken document. Once the ASR lexicon has been extended with content-relevant OOV proper nouns, those OOV words that were misrecognized in the ASR initial hypothesis could be recovered in second-pass decoding with the extended lexicon.

Our approach to recover OOV words infers the semantic context of a spoken utterance using the error-prone ASR initial hypothesis. Based on the inferred context, a list of likely relevant OOV proper nouns are added to the lexicon before performing second-pass decoding. Therefore, the primary goal of this thesis is to leverage word contexts in Wikipedia to develop a system that retrieves relevant OOV proper nouns given their in-vocabulary contexts. Then, this system can be used to extend the ASR lexicon and recover missing OOV words in ASR transcriptions.

1.4 Thesis Outline

This thesis is organized as follows:

- Chapter 2 gives an overview of some topics that are relevant to the problem is addressed in this thesis. It also presents a brief literature review of the OOV words problem in speech recognition.
- Chapter 3 gives a detailed description of the problem that addressed in this thesis, our proposed solution, as well as the evaluation procedure.
- Chapter 4 contains information about the data and presents experimental setup used in this research.
- Chapter 5 presents and discusses the results of the experiments that are conducted in this research.
- Chapter 6 concludes the thesis and suggests a few ideas for future work.

Chapter 2

Background and Literature Review

This chapter contains essential background information that is necessary to understand the methods developed in this thesis. Moreover, we present a brief literature review to highlight some research problems that are related to the topic of this thesis.

2.1 Automatic Speech Recognition

The problem of automatic speech recognition has been approached from an information-theoretic point of view with the noisy channel model (Shannon, 2001; Jelinek, Bahl, and Mercer, 1975). In this model, the sequence of words that corresponds to the spoken utterance is treated as a source signal that has passed through a noisy communication channel. Due to the noise in the channel, a distorted version of the source signal has been produced as an acoustic waveform. In order to recover the source signal from this distorted signal, we need to model the noise in the channel. Therefore, speech recognition is considered as a probabilistic inference problem where the goal is to find the most likely word sequence in the language given acoustic evidence. This goal could be formalised as a case of Bayesian inference as follows:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{A}) \tag{2.1}$$

where $\hat{\mathbf{W}} = \{\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_n\}$ is the hypothesized word sequence that corresponds to the spoken utterance, $\mathbf{A} = \{a_1, a_2, \ldots, a_T\}$ is a sequence of observations that constitute the acoustic evidence, $P(\mathbf{W}|\mathbf{A})$ is the probability of the word sequence given the acoustic evidence. Using Bayes's theorem, the equation can be rewritten as:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \frac{P(\mathbf{A}|\mathbf{W}) \times P(\mathbf{W})}{P(\mathbf{A})} \tag{2.2}$$

where $P(\mathbf{W})$ is the prior probability of the word sequence, $P(\mathbf{A}|\mathbf{W})$ is the likelihood of the observed acoustic evidence given the word sequence, and $P(\mathbf{A})$ is the probability of the observed acoustic evidence. Since the term $P(\mathbf{A})$ is independent of the word sequence $\mathbf{W}$, the expression can be simplified into:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{A}|\mathbf{W}) \times P(\mathbf{W}) \tag{2.3}$$

This formulation of the ASR problem in equation (2.3) is an instance of probabilistic generative modelling. The true distributions of $P(\mathbf{A}|\mathbf{W})$ and $P(\mathbf{W})$ are unknown, thus we need to estimate these probabilities from data. The ASR formulation can be decomposed into three main components:

- An acoustic model that estimates the likelihood of the observed acoustic evidence given a word sequence $P(\mathbf{A}|\mathbf{W})$. The parameters of this model are estimated from training datasets of spoken utterances that are annotated to build the statistical relationship between $\mathbf{W}$ and $\mathbf{A}$. The acoustic model is accompanied by a lexicon that maps each word into its phonetic pronunciation which can be obtained by human linguistic expertise or automatic grapheme-to-phoneme (G2P) converter. The lexicon is a finite word inventory that is often referred to as the pronunciation dictionary, and the ASR system can recognise only words within this dictionary.

- A language model that estimates the prior probability of the word sequence $P(\mathbf{W})$. The goal of the language model is to assess the fluency of the hypothesised sequence of words to make sure the words constitute a valid sentence in the target language. The parameters of the language model are estimated using large generic text corpora.

- A decoding algorithm to perform the search formalised by the term $\arg\max_{\mathbf{W}}$. The goal of the decoding algorithm is to find the word sequence that maximises the *posteriori* probability by searching over all possible word sequences in the target language. Because the search space over all possible word sequences is exponentially large, decoding algorithms are based on optimisation techniques and dynamic programming to alleviate the complexity of the search.

Each spoken utterance is usually associated with a single ground-truth reference transcription. Therefore, evaluating the performance of ASR systems is a straightforward text matching task between the ASR hypothesis and the reference transcription. The standard metric for evaluating ASR systems is the *Word Error Rate* (WER) metric that is defined as follows:

$$WER = \frac{D + I + S}{N} \tag{2.4}$$

where D is the number of deletions (i.e., words that are omitted in the ASR hypothesis), I is the number of insertions (i.e., words that are added to the ASR hypothesis), S is the number of substitutions (i.e., words that are replaced by other words the ASR hypothesis), and N is the total number of words in the ground truth reference.

2.2 Distributed Representations of Words and Documents

In the early days of statistical natural language processing (NLP), a word in the language lexicon was viewed as an atomic unit of meaning. This view results in discrete word representations where each word is equally distant from the other words in a high-dimensional

space. Therefore, those discrete representations do not capture semantic similarity and relatedness between lexical units. NLP systems that are trained on such discrete representations cannot generalise when they have to deal with contexts that are not observed in the training data. For example, if the word *camel* has been observed after the context *feed a* ___ in a text corpus, an n-gram language model trained on the same corpus would often give a high probability to the word *camel* when estimating the probability $P(camel|feed\ a)$. However, if the word *llama* has never observed after the same context, it will be given much lower probability than the word *camel* even though the two continuations are equally plausible. This limitation of statistical n-gram language models, which are based on frequency statistics, is a result of the poor generalisation of conventional count-based NLP systems.

To enable NLP systems to make better generalisations beyond what has been observed during training, different techniques to estimate word representations in a continuous-space have been proposed in the literature (Turney and Pantel, 2010; Mitchell and Lapata, 2010; Mikolov et al., 2013; Pennington, Socher, and Manning, 2014). All these techniques are motivated by the distributional perspective of lexical semantics which hypothesises that word meaning is always contextual and words with similar meaning have similar context distributions (Harris, 1954). This perspective of inducing word representations from contexts makes it possible to construct a semantic space where conceptually similar words like *camel* and *llama* are nearby if their contexts are similar in the corpus.

Estimating task-independent generic word representations by exploiting contextual information using neural networks has received sizable attention from the NLP community in the last few years. The most notable work in this direction is Mikolov et al. (2013) which proposed several novel log-linear models and efficient estimation algorithms to estimate low-dimensional distributed word representations, also known as word embeddings[1], from raw text. The resulting word representations from Mikolov et al.'s models are real-value vectors ($\mathbf{v}_w \in \mathbb{R}^d$) learned as parameters of the model during training. Each dimension in the d-dimensional space can be viewed as a latent feature that encodes semantic or syntactic information about the word. Mikolov et al.'s work has stimulated a new line of research dedicated to word representation models using neural techniques.

In this chapter, the continuous skip-gram model of Mikolov et al. (2013) is introduced due to its relevance to the research that is conducted in this thesis. The continuous skip-gram is a log-linear model that is trained with the objective of maximising the probability of context words given a target word. The training procedure of this model iterates over each word in a large text corpus and aims to predict the words that occur in a fixed context window. This objective can be formalised as maximising the probability of the context words given the target word. Figure 2.1 illustrates the training procedure of the skip-gram model. The size of the context window is a hyperparameter of the model. It has been observed that large context window results in word vectors that encode more semantic or topical features (as opposed to syntactic features). The authors have released an efficient open source implementation of their models that is well-known as `word2vec`.

[1] In this thesis, the terms *word representations* and *word embeddings* are used interchangeably to refer to the same concept; words vectors in a low-dimensional continuous space.

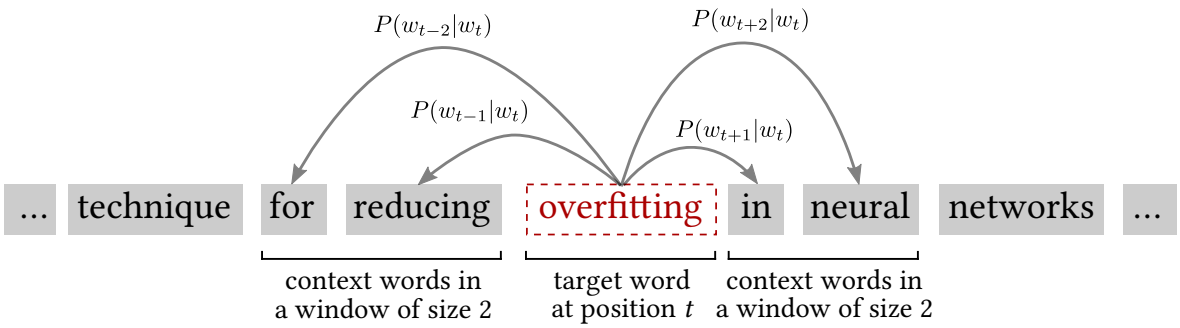

FIGURE 2.1: An illustrated example of the training procedure of the continuous skip-gram model; given a target word, maximise the probability of context words.

Furthermore, it has been observed that word representations learned by the continuous skip-gram model exhibit compositional properties. For example, if we perform element-wise addition of the two vectors $\mathbf{v}_{German}$ and $\mathbf{v}_{airlines}$, the nearest neighbour in the semantic space to this summed vector would be $\mathbf{v}_{Lufthansa}$. This property could be expressed by the following vector operation:

$$\mathbf{v}_{German} + \mathbf{v}_{airlines} \approx \mathbf{v}_{Lufthansa} \tag{2.5}$$

This additive compositional property shows that word representations learned by the skip-gram model exhibit some linear structure that makes it possible to combine words to obtain another unit of meaning by simple element-wise vector addition. The reason behind this compositional property is related to the training objective of the skip-gram model. This model is trained to maximise the probability of context words for a given central target word; thus the vectors reflect the contexts in which the target word occurs nearby. For example, the contexts distribution of the word *Lufthansa* could be viewed as the intersection of the contexts nearby the words *German* and *airlines*. This finding indicates that the statistical observations about word contexts are encoded within the word embeddings as a linear structure that could be exploited by algebraic vector operations to form other units of meanings.

The property of additive compositionality has been an interest to researchers within the information retrieval (IR) community. In IR, one fundamental task is to retrieve a relevant document in response to a query; both are expressed as a sequence of words (although a query usually consists of a few words while documents are much larger). In order to retrieve relevant documents, there is a need to quantify the similarity between the query and each document in the document collection. The standard approach of quantifying the similarity between text fragments is by using the cosine distance measure over their vector representations. To obtain a single vector representation for a text fragment that consists of a sequence of words (e.g., document, sentence, etc.), some researchers proposed to average the word embeddings (Kiros, Zemel, and Salakhutdinov, 2014; Kenter, Borisov, and Rijke, 2016). A vector representation of a document d can be obtained by averaging the word

embeddings as follows:

$$\mathbf{d} = \frac{1}{|d|} \sum_{w \in d} \mathbf{v}_w \tag{2.6}$$

For information retrieval tasks, Mitra and Craswell (2017) proposed cosine similarity as a scoring function for ranking documents according to their relevance to a query. Given vector representations of a query $\mathbf{q}$ and a document $\mathbf{d}$, the cosine similarity is defined as:

$$sim(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\|^2 \cdot \|\mathbf{d}\|^2} \tag{2.7}$$

where $\cdot$ is the vector dot product operation, and $\|\mathbf{x}\|^2$ is L2 norm of the vector $\mathbf{x}$. In this thesis, we used a similar approach in order to rank relevant OOV words to the averaged word embeddings of a transcribed spoken document.

2.3 Text Classification with Neural Networks

Some aspects of the work presented in this thesis rely on neural text classification techniques. Therefore, this section presents the methods that our work was built on. In NLP, text classification, or categorisation, is the task of assigning a label in a predefined set of k labels to a text document represented as a sequence of word tokens W. If a document can strictly be assigned to a single label, this is a case of multi-class classification. For example, a movie review can either be positive or negative regarding the movie. If a document can be assigned to more than one label in the k-label set, this problem is referred to as multi-class multi-label classification. An example of multi-class multi-label classification is assigning a subset of predefined categories to a Wikipedia article. In this section, we present three common neural techniques for multi-class text classification. In chapter 3, we describe how these techniques can be adapted for multi-class multi-label problems.

2.3.1 Deep Averaging Networks

A deep averaging network (DAN) is a feed-forward neural network with one or more hidden layers that has been proposed for text classification problems (Iyyer et al., 2015). In this model, each document is represented by the averaged word embeddings of its word tokens; hence the order of the words in the document is not preserved. This averaged vector of a word sequence W can be expressed as follows:

$$\mathbf{z}_0 = \frac{1}{|W|} \sum_{w \in W} \mathbf{v}_w \tag{2.8}$$

Then, the averaged vector is passed through one or more feedforward layers in order to learn higher-level representations of the input. If the input representation has to pass through

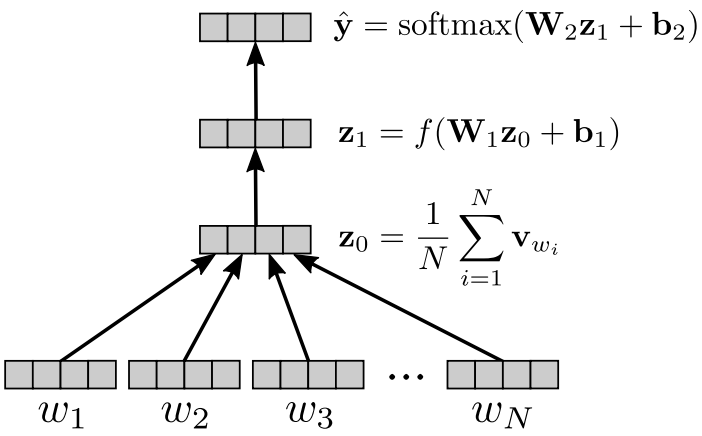

FIGURE 2.2: Deep Averaging Network with one hidden layer.

n layers, the computation at layer i is the following:

$$\mathbf{z}_i = f(\mathbf{W}_i \cdot \mathbf{z}_{i-1} + \mathbf{b}_i) \tag{2.9}$$

Here, $f(\cdot)$ is a non-linear activation function (e.g., sigmoid) parametrised by weight matrix $\mathbf{W}_i$ and bias vector $\mathbf{b}_i$. Finally, an output layer is used to estimate a probability for each label in the label space, which could be a softmax that takes as an input the representation computed as layer n:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_n \cdot \mathbf{z}_{n-1} + \mathbf{b}_n) \tag{2.10}$$

Despite its simple architecture, it has been shown that this model gives comparable performance to many complex models in many text classification benchmarks (Iyyer et al., 2015). A schematic view of DAN is shown in Figure 2.2.

2.3.2 Convolutional Neural Networks

Although convolutional network networks (CNNs) have been initially designed for classification problems in computer vision (LeCun and Bengio, 1995), they have been adapted for text classification problems in NLP (Collobert et al., 2011; Kalchbrenner, Grefenstette, and Blunsom, 2014; Kim, 2014; Zhang and Wallace, 2015). To apply convolutional neural networks for text classification, a document that consists of N-word sequence $(w_1, w_2, \ldots, w_N)$ is represented as a matrix by arranging the d-dimensional word embeddings as a $d \times N$ matrix as follows:

$$\mathbf{v}_{1:N} = \begin{bmatrix} | & | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_N \\ | & | & | & | \end{bmatrix} \tag{2.11}$$

In this notation, for example, a trigram is viewed as the arrangement of the word embeddings of three adjacent words in the following submatrix structure:

$$\mathbf{v}_{i:i+2} = \begin{bmatrix} | & | & | \\ \mathbf{v}_i & \mathbf{v}_{i+1} & \mathbf{v}_{i+2} \\ | & | & | \end{bmatrix} \tag{2.12}$$

A representation of a higher-order m-gram is a generalization of the expression in 2.12:

$$\mathbf{v}_{i:i+m-1} = \begin{bmatrix} | & | & | & | \\ \mathbf{v}_i & \mathbf{v}_{i+1} & \dots & \mathbf{v}_{i+m-1} \\ | & | & | & | \end{bmatrix} \tag{2.13}$$

Since languageh is temporal in nature (i.e., change occurs only in the time dimension), a one-dimensional convolutional operation can be applied to extract high-level features that are relevant to the prediction task from the distributed word representations. To achieve this, a convolutional filter $\mathbf{F}_{conv} \in \mathbb{R}^{m \times d}$, where d is the word embedding dimension and m is the window size, is applied to a window of m words to obtain a new feature as:

$$c_i = f(\mathbf{F}_{conv} \bullet \mathbf{v}_{i:i+h-1} + b) \tag{2.14}$$

where c_i is a scalar, $\mathbf{F}_{conv}$ and b are trainable parameters that are shared across time (i.e., same set of parameters applied to all possible windows $\mathbf{v}_{i:i+m-1}$), the $\bullet$ operation is an element-wise matrix multiplication followed by summation (in order to obtain a scalar), and f is a non-linear activation function. The parameters of the filter $\mathbf{F}_{conv}$ are learned when training the neural network and back-propagating the error in a supervised learning task. The convolutional filter is then applied to all successive m-size windows by sliding the filter over the $d \times N$ matrix that represents the input document to generate a feature map as follows:

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N-h+1} \end{bmatrix} \tag{2.15}$$

where $\mathbf{c} \in \mathbb{R}^{N-m+1}$. After that, max-over-time pooling operation is applied over the feature map to extract the element with the maximum value as the feature that is captured by the filter:

$$\hat{c} = max(\mathbf{c}) \tag{2.16}$$

The max pooling operation makes it possible to obtain fixed size representations for variable-length sequences. A convolutional architecture for text classification consists of multiple filters (usually with various window sizes) in order to generate multiple features. The concatenation of all generated features forms a vector that captures the feature representation of

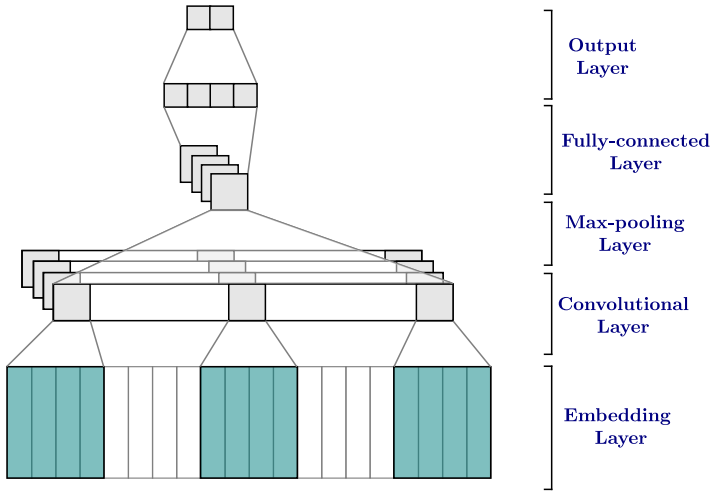

FIGURE 2.3: Convolutional Neural Network for text classification.

the input document. The vector representation is then passed to a fully-connected softmax layer to obtain a probability distribution over the set of output labels. A schematic view of a text classification CNN is shown in Figure 2.3.

2.3.3 Recurrent Neural Networks

In neural networks, the conventional architecture to process variable-length sequences is a recurrent neural network (RNN). Given a sequence of words represented by their embeddings $\langle \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N \rangle$, an RNN is recursively defined as follows:

$$\mathbf{h}_t = \begin{cases} f_{RNN}(\mathbf{h}_{t-1}, \mathbf{v}_t; \theta) & \text{if } t \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

This recursive definition enables the RNN to process sequences of arbitrary length, and that is the case with natural language utterances. In theory, the hidden state $\mathbf{h}_t$ is a summary of the information that has been observed in the sequence up to timestep t. In practice, vanilla RNNs fails to encode all information in the hidden state when sequences are long. It was shown that this limitation of RNNs is caused by the vanishing gradient problem (Bengio, Simard, and Frasconi, 1994; Jozefowicz, Zaremba, and Sutskever, 2015). To address this problem, gated recurrent structures with memory elements have been proposed. The most notable variants of gated recurrent structures are the long-short term memory (or LSTM, proposed by Hochreiter and Schmidhuber, 1997) and the gated recurrent unit (or GRU, proposed by Cho et al., 2014). Since we use GRUs in this thesis, we introduce the typical computational dataflow for a GRU in this section. A GRU is characterized by two gates; reset gate $\mathbf{r}_t$ and update gate $\mathbf{z}_t$. The motivation to use these two gates is to adaptively control how much information from the hidden state should be carried to the next state and how much information from the current input should be taken in the next hidden state. The vectorized computation of GRU can be expressed as follows:

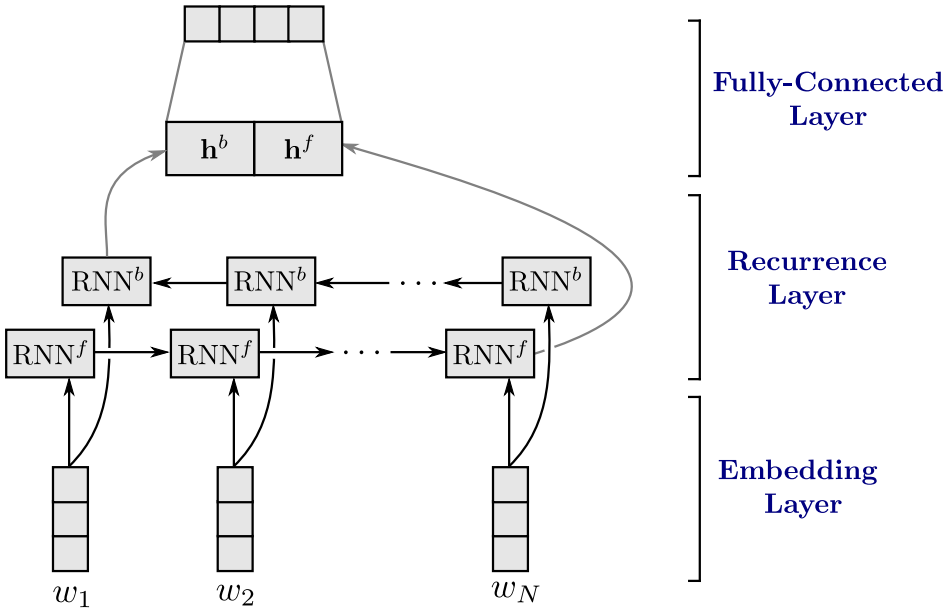

FIGURE 2.4: Recurrent Neural Network for text classification.

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{v}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \tag{2.17}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{v}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \tag{2.18}$$

$$\widetilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{v}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \tag{2.19}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \widetilde{\mathbf{h}}_t \tag{2.20}$$

The recurrent computation presented so far considers processing the input sequence in one direction. The last hidden state in the RNN can be considered as a representation that compresses the entire input sequence in a vector. However, it was shown that processing the sequence in two directions (i.e., forward and backward) can improve the performance of many sequence modelling applications such as machine translation (Bahdanau, Cho, and Bengio, 2014) and speech recognition (Graves, Mohamed, and Hinton, 2013). In the bidirectional recurrent structure, the last hidden states in the forward direction and backward direction are merged (either by concatenation or element-wise addition) to form a single vector that represents the sequence. For text classification tasks, this vector is transformed via a fully connected layer into softmax vector that represents a probability distribution over the output space. A schematic view of a bidirectional RNN for text classification is depicted in Figure 2.4.

2.4 Related Work

Since accurate recognition of proper nouns is vital for spoken utterance understanding, the OOV words problem has received sizable attention from the speech recognition community.

Previous approaches that deal with this problem can be broadly grouped into two main categories:

1. Approaches that attempt to detect the presence of OOV words by locating potential OOV regions in the hypothesised transcription of the ASR. These approaches can be referred to as OOV detection-based approaches.

2. Approaches that leverage some knowledge about the OOV word co-occurrence statistics with in-vocabulary words in order to extend the lexicon of the ASR system. These approaches can be referred to as OOV recovery-based approaches.

A brief introduction to these two categories and notable works in each category is presented here.

2.4.1 OOV Words Detection

Detecting the presence of OOV words in a speech transcription can be tackled as a sequence labelling problem, without assuming any prior knowledge of OOV words and their statistical co-occurrence patterns. Given the first-pass hypothesis of an ASR system, the OOV words detection problem can be approached as a binary classification task over the words. That is, the task is to decide whether each word in the hypothesised transcription is an OOV that has incorrectly transcribed or in-vocabulary (IV) word, given some feature representation of that word. In this line of research, Lecouteux, Linares, and Favre (2009) have proposed representing each word as a feature vector that consists of 23 different features. These features can be grouped into 3 different categories: (1) acoustic features that are induced from the likelihood scores of the acoustic model, (2) linguistic features that are based on the probabilities of the language model integrated into the ASR, (3) graph features that are motivated by the mechanism of the decoding algorithm in ASR and the word confusion networks. The idea of representing words in the ASR hypothesis as feature vectors encourages the use of various hand-crafted features into the classification problem. Thus, many features can be engineered and their effectiveness could be investigated. Nevertheless, even if the presence of an OOV word is located within the hypothesis, it is not trivial to recover the orthographic form of the OOV word from the in-vocabulary word(s) that substituted it.

Another approach to detecting OOV word regions is by representing OOV words as sub-word units such as phones or syllables. In this approach, OOV words in a test spoken utterance are going to be recognised as sequences of sub-word units instead of misrecognized in-vocabulary words. This approach requires a hybrid language model and a hybrid lexicon, which are trained on both the words as well as the sub-word units, to be integrated into the ASR framework during decoding phase. One work in this direction is the work of Klakow, Rose, and Aubert (1999) which proposed the use of a flat hybrid system that models words and sub-word units. To train their system, the authors used the most frequent 5000 words as in-vocabulary words and treated all words outside this list as pseudo-OOV words. Sub-word models are trained using the phonetic transcriptions of the pseudo-OOV words in hybrid the lexicon. Then the occurrence of each pseudo-OOV word in the training

data was substituted by its corresponding sub-word units. A hybrid language model was trained on this combination of words and sub-word units.

2.4.2 OOV Words Recovery

Contrary to OOV detection based approaches where OOV words are not known beforehand, OOV recovery approaches attempt to exploit some source of external knowledge in order to infer the presence of OOV words in a spoken utterance. Information from this source of external knowledge could be incorporated in the ASR system to recover OOV words. In its simplest form, the knowledge source could be a predefined set of OOV words that are relevant to the domain of the spoken utterances to be transcribed (e.g., names of cities and provinces in a weather forecast reports).

One source of knowledge that can be exploited for OOV words recovery is the World Wide Web. Co-occurrence statistics between in-vocabulary words and OOV words can be estimated using text documents in the web. In this direction, the work of Parada et al. (2010) has proposed an approach that first detects OOV regions within the first-pass hypothesis of the ASR system then uses the lexical context of the detected OOV regions to query the web using an off-the-shelf search engine (e.g., Google). The lexical context consists of words that are in the neighbourhood of the detected OOV region in the transcribed utterance. The TF-IDF was used as a metric to quantify the importance of each word to a given document. The authors reported a recall of 40% of OOV words recovery and a reduction in the WER of the ASR. This work has shown that contextual information on the web could be exploited to recover OOV words by using a simple non-parametric text matching between the lexical context of the OOV region and online text documents. Thus, it would be interesting to explore and investigate whether this search procedure could be parametrised using the web as a corpus for OOV recovery.

The research conducted in this thesis is built on ideas proposed by Sheikh et al. (2017). The authors rely on the global context obtained from the first-pass ASR hypothesis to predict and recover the OOV words that are likely to occur in that context, without the need to detect OOV regions in the ASR transcription. To this end, a corpus of diachronic news articles, which was collected by crawling the web, was used to obtain semantic contexts of OOV words. Two different approaches were used to build context representations and predict relevant OOV words: (1) Latent Dirichlet Allocation (LDA) topic modelling methods to induce unsupervised context representations, (2) discriminative neural network-based models to build up context representations in a supervised manner. It has been observed that supervised neural models outperform LDA-based context representations for retrieving relevant OOV words. The experiments of Sheikh et al. (2017)'s work indicate that models built using deep neural networks could give further improvements on the retrieval performance. One of the main contributions of Sheikh et al. (2017) was exploiting diachronic text corpora collected from the web to build parametric models that could predict the presence of OOV words in a given spoken utterance.

Chapter 3

Proposed Approaches

3.1 Problem Description

The research conducted within this thesis is concerned with a single component in a larger ASR system that is designed to recover missing OOV proper nouns in a hypothesised transcription. The overall procedure of the system can be described in the following steps:

1. Given a spoken utterance, the ASR is run to obtain an automatic transcription of the utterance (i.e., first-pass hypothesis)
2. Based on the context of the first-pass hypothesis, which is a sequence of in-vocabulary words, a list of likely relevant OOV proper nouns (PNs) is produced
3. A grapheme-to-phoneme (G2P) conversion is applied to get pronunciations for each word in the proposed list in step (2)
4. The relevant OOV PNs and their corresponding pronunciations are added to the lexicon
5. Language model probabilities are estimated for relevant OOV PNs
6. A second-pass ASR decoding is performed using the extended lexicon with the retrieved OOV PNs

Figure 3.1 illustrates the framework of an ASR system with a recovery mechanism for OOV words. Successful recovery of OOV proper nouns in the second-pass decoding depends on the performance of each component in the steps mentioned above. For step 3, phonetic pronunciations for OOV proper nouns can be obtained using a grapheme-to-phoneme (G2P) conversion system that has been previously developed (Illina, Fohr, and Jouvet, 2011). Estimating language model probabilities for new words in the extended lexicon (step 5) has also been addressed in a previous work (Currey, Illina, and Fohr, 2016). In this thesis, we approach the problem of retrieving relevant OOV proper nouns (step 2). Therefore, this thesis aims to develop parametric models that infer the semantic context of a spoken utterance from the first-pass hypothesis, then propose a list of OOV proper noun candidates that are likely to occur in that context. We refer to the OOV retrieval system as a *context model*.

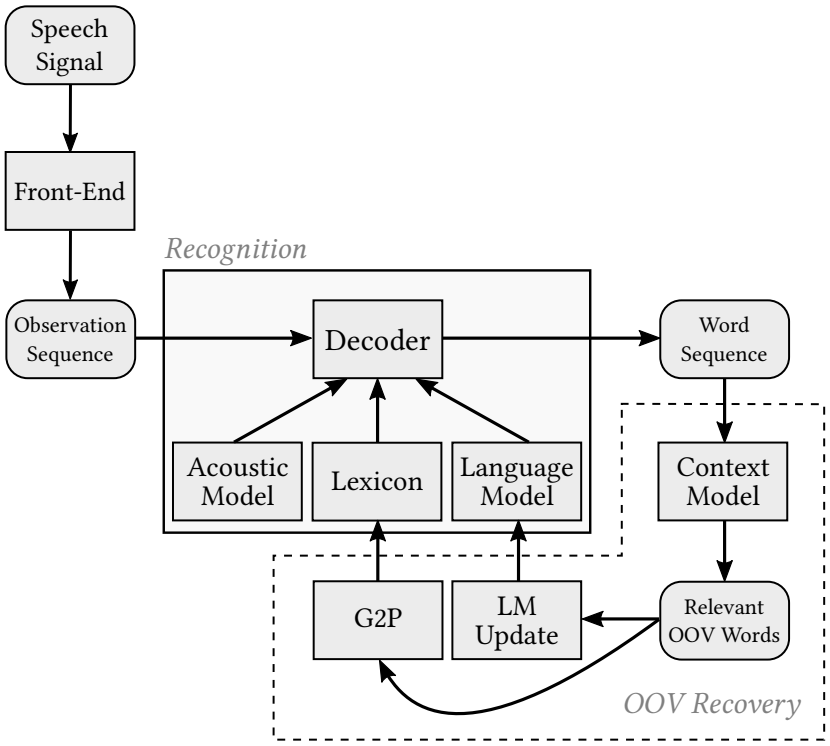

FIGURE 3.1: Framework of the extended ASR system with an OOV words recovery mechanism. Components within the dashed area are used to dynamically extend the ASR lexicon before performing a second-pass decoding.

3.2 Proposed Methodology

This section gives a detailed description of the proposed methods to add OOV proper nouns to the ASR lexicon, given a hypothesised transcription. As discussed earlier, ASR systems are trained on a finite word vocabulary in a fixed lexicon. Therefore, ASR systems cannot correctly transcribe words that do not exist in the lexicon. Context-independent extension of the ASR lexicon based on word frequency statistics does not necessarily result in better performance nor better recognition of proper nouns for two main reasons:

- A larger lexicon would make the decoding search space more complex. Hence, ASR performance usually degrades when the lexicon incorporates words that are irrelevant to the spoken context.
- The majority of OOV proper nouns are rare words. Thus, a context-independent extension of the ASR lexicon is very unlikely to succeed in including proper nouns that are actually relevant to every spoken document during testing.

To address these problems, we propose a document-specific approach to extend the ASR lexicon based on the global context obtained from the first-pass hypothesis. That is, the base vocabulary of the ASR system is kept fixed during training. During inference, the ASR lexicon is extended based on the context of each spoken document, which can be inferred from the semantic information in ASR first-pass hypothesis. Given the inferred context, the

goal is to retrieve highly relevant OOV proper nouns and perform second-pass decoding. To this end, we need to solve two sub-problems:

- How to obtain a list of OOV proper nouns with good coverage over the spoken language? That is, we need to have a large inventory of OOV proper nouns at our disposal to recover missing words in ASR transcriptions.
- How to model the relationship between in-vocabulary words and OOV proper nouns? That is, we need to retrieve missing OOV proper nouns in an ASR hypothesis by performing inference over a context of in-vocabulary words.

To solve these two subproblems, we propose to use a large text corpus collected from the web. On the one hand, a large text corpus usually provides a good lexical coverage over the language hence a large inventory of OOV proper nouns can be obtained. On the other hand, we can capture the relationship between in-vocabulary words and OOV proper nouns using the contextual information in the text corpus.

In this thesis, we do not aim to identify OOV regions in the ASR hypothesis. Instead, we rely entirely on the global context of each document to retrieve OOV proper nouns. We argue that if the relevant OOV proper nouns are retrieved by the context model and their phonetic pronunciations are added to the ASR system, proper nouns that actually occur in the spoken document can be successfully transcribed in second-pass decoding without the need to locate OOV regions. The advantages of our approach are twofold. First, the ASR lexicon is dynamically adapted to each spoken document without the need to retrain the other components of the system. Second, the orthographic forms of OOV proper nouns are known beforehand. Thus, it is possible to recover target proper nouns in second-pass decoding if they are successfully retrieved.

To describe the problem formally, let $\mathcal{V}$ be the *base vocabulary* of the ASR system such that $\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{V}|}\}$, where each v_i is an in-vocabulary word. From a large text corpus, a set of OOV proper nouns denoted $\mathcal{V}_{OOV}$ can be extracted. The task is to build a system that takes as an input a sequence of in-vocabulary words $W = \langle w_1, \ldots, w_N \rangle$ and outputs a set of OOV proper nouns $\tilde{\mathcal{V}} = \{v_{OOV_1}, \ldots, v_{OOV_M}\}$, such that $\tilde{\mathcal{V}} \subset \mathcal{V}_{OOV}$ and $M \ll |\mathcal{V}|$. In our setting, this task could be further formalised as follows:

$$\tilde{\mathcal{V}} = f(W) \tag{3.1}$$

where $\tilde{\mathcal{V}} \subset \mathcal{V}_{OOV}$ is a hypothesised set of OOV proper nouns that are likely to be relevant to the in-vocabulary word sequence W. In its simplest form, the function f could be a non-parametric procedure that takes text segments from the ASR hypothesis and generates queries for a web search engine (e.g., Google). Then, the retrieved documents by the search engine are processed to extract OOV word candidates. Our goal in this thesis is to explore methods to parametrise the function f as:

$$\tilde{\mathcal{V}} = f(W; \theta) \tag{3.2}$$

In other words, the goal is to find a set of parameters θ such that the OOV proper nouns retrieval performance given the input W is optimal. To be more specific, we explore neural architectures in NLP to model the function f and leverage a large text corpus to learn the parameters of the proposed model. The idea is that if we build a model that predicts the presence of OOV proper nouns from their in-vocabulary contexts, this model can be used to recover missing OOV words in ASR transcriptions.

One problem with this formulation of the task is the vast output space, that is, the size of the OOV proper nouns inventory. For example, there are approximately 1.8 million OOV proper nouns in the French Wikipedia (more information about the data used in this study is provided in Chapter 4). Compared to the ASR base vocabulary in our setting (∼97k words), recommending a ranked list of OOV proper nouns from 1.8M possible candidates is not trivial. Therefore, we need to scale down the problem and address the complexity associated with the output space. In this thesis, we address this complexity by proposing an intermediate step that first infers the class of OOV proper nouns from the context. That is, instead of one-step inference of OOV proper nouns over ASR hypothesised context (as formalised in equation 3.1), we propose a two-step approach as follows:

STEP 1 Predict the OOV PN *classes* that could occur in the context W, instead of directly predicting the OOV PNs

STEP 2 Rank the OOV PNs in the predicted classes based on their *relevance* to the spoken context to obtain $\tilde{\mathcal{V}}$

STEP 1 and STEP 2 are two related, yet different problems that we tackle in this thesis. In particular, we aim to solve these problems using methods based on deep neural networks. For STEP 1, we propose to organise OOV proper nouns into clusters and consider each cluster as a target class to predict. That is, each OOV proper noun is mapped into a single class where semantically similar OOV proper nouns are members of that class. This step could be formally described as follows:

$$\hat{\mathcal{C}} = f(W; \theta_C) \tag{3.3}$$

where $\hat{\mathcal{C}} \subset \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ is a set of OOV PN classes that are likely to be relevant to the context. To obtain a finite set of OOV PN classes, we use K-means clustering to group word representations into clusters (section 3.3). Since a spoken document could contain OOV proper nouns of different semantic types, a document can be mapped into several OOV classes. Therefore, we approach this task as a multi-class multi-label text classification problem (section 3.4).

For STEP 2, we propose a ranking procedure that takes as an input the ASR transcription of a spoken document, the OOV classes predicted in STEP 2, and the inventory of OOV proper nouns $\mathcal{V}_{OOV}$, then outputs a ranked list of OOV proper nouns $\tilde{\mathcal{V}}$ that will be added to the ASR lexicon before running a second-pass decoding. The ranking procedure can

formally be expressed as follows:

$$\tilde{\mathcal{V}} = g(W, \hat{\mathcal{C}}, \mathcal{V}_{OOV}) \tag{3.4}$$

Our ranking procedure is based on the cosine distance between the average word embeddings in the input (ASR hypothesis) and each OOV PNs in the reduced candidate list (section 3.5). The goal of the ranking step is to make sure that OOV proper nouns that are more likely to be relevant to the in-vocabulary ASR transcription are ranked higher in the candidate list $\tilde{\mathcal{V}}$ than other OOV proper nouns.

3.3 OOV Words Clustering

Many neural word representation models have been proposed in the literature during the last decade. The goal of these models is to build up a continuous-space representation for each word in the vocabulary by exploiting contextual information in a large text corpus. It has been observed that representations produced by the skip-gram model of Mikolov et al. (2013) encode both semantic and syntactic information about each word. In our task, we require word vectors such that words that tend to co-occur in the same topic would have similar vectors. To obtain word representations that encode more semantic than syntactic information about words, a large context window should be preferred. For the skip-gram model, the objective is to maximise the probability of context words given a target word. If the context window is large, the context words would usually be words that co-occur in the same topic with the target word. Thus, we use a context window of 20 words to build semantic word representations where words are nearby in space if they exhibit topical similarity.

In our task, we are interested in scaling down the output space from the set of all possible OOV proper nouns to a set of OOV clusters, where each cluster consists of semantically similar words. The motivations behind OOV words clustering are twofold. First, it is computationally infeasible to build models whose output is a probability distribution over hundreds of thousands of possible labels. Second, many OOV proper nouns are very rare and not well-represented in textual corpora. Thus, it is better to group infrequent OOVs with other semantically similar OOV words.

In order to group words into clusters, we use K-means clustering. K-means is a popular clustering algorithm that aims to partition T data points into K clusters. In our problem, the data points are continuous-space word vectors $\{\mathbf{v}_{w_1}, \mathbf{v}_{w_2}, \ldots, \mathbf{v}_{w_T} | \mathbf{v}_{w_i} \in \mathbb{R}^d\}$. We use the K-means clustering algorithm to partition the set of T word vectors into K sets $\mathbf{C} = \{c_1, \ldots, c_K\}$ such that the distance [1] between each word vector and its cluster centroid is minimal. This

[1] Euclidean distance is presented as the measure in this section. However, other measures of vector distance can be used (e.g., cosine distance)

objective can be formally expressed as follows:

$$\arg\min_{\mathbf{C}} \sum_{i=0}^{T} \sum_{j=0}^{K} \left\| \mathbf{v}_{w_i}^{(j)} - c^{(j)} \right\|^2 \tag{3.5}$$

where $c^{(j)}$ is the centroid for cluster j. Given this objective, the resulting clusters are word clusters where semantically similar OOV words are grouped within the same cluster. A pseudo-code for K-means is presented in algorithm 1.

INPUT : $\mathbf{V} = \{\mathbf{v}_{w_1}, \mathbf{v}_{w_2}, \ldots, \mathbf{v}_{w_T} \mid \mathbf{v}_{w_i}\}$ (set of word vectors to be clustered)
K (number of clusters)
maxIter (maximum number of iterations)
OUTPUT: $\mathbf{C} = \{c_1, \ldots, c_K\}$ (cluster centroids)
$M = \{m(\mathbf{v}_{w_i}) | \; m(\mathbf{v}_{w_i}) \in \mathbf{C}\}$ (word vector to cluster dictionary)

```
for i ← 1 to K do
    c_i ← v_{w_j} ∈ V        (random initialization)
end
for i ← 1 to T do
    m(v_{w_i}) ← arg min_c ||v_{w_i} − c||^2        (assign vectors to initial clusters)
end
iter ← 1
while iter < maxIter do
    for i ← 1 to K do
        updateCentroid(c_i)        (update step)
    end
    for i ← 1 to T do
        min ← arg min_c ||v_{w_i} − c||^2
        if min ≠ m(v_{w_i}) then
            m(v_{w_i}) ← min        (assignment step)
        end
    end
    iter++
end
```

Algorithm 1: K-means Clustering

K-means starts by randomly selecting K data points as initial centroids, then each data point is assigned to its nearest centroid. The two main iterative procedures in K-means clustering is the centroid update procedure and cluster assignment procedure. The centroid update procedure computes new means to be centroids of the clusters in the next iteration. The cluster assignment procedure moves the data point to the cluster with the nearest centroid. These two procedures are run for a number of iterations either as specified by the input *maxIter* or until the centroid update procedure converges (that is, centroids stop moving). The main output of K-means clustering in our setting is a data structure M that can be viewed as a dictionary that maps a word to its cluster ID. The choice of the K used in this study is discussed and motivated in Chapter 4. A few examples of the produced clusters are

presented in Chapter 5.

3.4 OOV Class Prediction

Given a large collection of text documents, we can obtain an inventory of OOV words $\mathcal{V}_{OOV}$ and their in-vocabulary contexts. A text document is a sequence of word tokens $\langle w_1, \ldots, w_N \rangle$. Each word token in the sequence is either a word that exists in the ASR lexicon ($w_i \in \mathcal{V}_{IN}$) or an OOV proper noun ($w_i \in \mathcal{V}_{OOV}$). The goal of this thesis is to build models that take as an input a sequence of in-vocabulary words (without OOVs) and output a set of OOV proper nouns that are likely to occur in that context. After that, these models are used to retrieve missing proper nouns in ASR transcriptions.

As discussed earlier in this chapter, the number of OOV words is substantial (for example, 1.8M OOV proper nouns in Wikipedia). Therefore, we propose to scale down the problem into predicting classes of OOV words. We apply K-means clustering on the word embeddings to obtain these classes. However, a spoken or written document usually covers a mixture of topics. Thus, a single document might contain OOV words from different classes. Mapping a document to a one or more classes is a case of the multi-class multi-label classification problem. An example of this task is assigning a Wikipedia article to a set of predefined categories that organise the topical content in Wikipedia. We address the problem of classifying a document with respect to the OOV classes using a similar approach.

3.4.1 Training

In multi-class multi-label text classification problems, we are given a training dataset of n tuples $\mathcal{D} = \{(X^{(1)}, Y^{(1)}), \ldots, (X^{(n)}, Y^{(n)})\}$ where $X^{(i)}$ is a sequence of word tokens and $Y^{(i)}$ is a set of ground-truth labels. The predefined set of possible labels could be large. In our task, we do not have a manually labelled data with respect to the OOV classes. Therefore, we set the predefined labels to be the clusters induced by applying K-means clustering on the word vectors.

To generate training data for our task, any generic text corpus can be used as long as it is large enough to cover infrequent OOV words. Then each document d in the corpus that contains OOV proper nouns is turned into a training instance as follows:

1. All in-vocabulary words $w_i \in \mathcal{V}$ are extracted according to their order in the document to create a word sequence $X = \langle w_1, \ldots, w_M \rangle$
2. All OOV proper nouns $w_i \in \mathcal{V}_{OOV}$ are extracted, and each OOV proper noun is mapped to its corresponding cluster $C_i \in \mathbf{C}$
3. The set of clusters $Y = \{C_1, \ldots, C_L\}$ obtained in step (2) constitute the labels of this training instance
4. A training instance is the tuple (X, Y)

Note that the frequency of an OOV class in a document is not taken into account, but its presence. For example, if a document contains 20 occurrences of OOVs that belong to class C_a, and a single OOV occurrence of class C_b, both C_a and C_b are considered equally as labels.

Once the training data is created, any of text classification methods described in section 2.3 can be used. In a neural network, each input document X is transformed into a continuous-space representation by one or more layers as follows:

$$\mathbf{z} = \phi(X) \tag{3.6}$$

In this thesis, we explore two types of neural models to build up the continuous document representation $\mathbf{z}$: (1) unordered averaging models that represent the document as the average of the embeddings of its words, and (2) sequence models that take into account the sequential nature of the text data to build the document representation (precisely, we explore recurrent and convolutional sequence modeling architectures). To obtain the probability of each class, the document representation $\mathbf{z}$ is transformed via a fully-connected output layer. For multi-class multi-label classification, the output layer of the neural network is a sigmoid layer as follows:

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_o \cdot \mathbf{z} + \mathbf{b}_o) \tag{3.7}$$

where $\hat{\mathbf{y}} \in \left[0,1\right]^K$ and K is the number of OOV classes. To train the neural network in a multi-class multi-label classification setting, a binary cross-entropy loss function is used. The cross-entropy loss of a single training instance can be expressed as follows:

$$\mathcal{L}(\theta) = -\mathbf{y} \cdot log(\hat{\mathbf{y}}) \tag{3.8}$$

The overall objective of the training procedure is to find the set of parameters θ such that the overall loss computed over the n training instances is minimum:

$$\mathcal{J}(\theta) = -\sum_{i=1}^{n} \mathbf{y}^{(i)} \cdot log(\hat{\mathbf{y}}^{(i)}) \tag{3.9}$$

Then, the error signal obtained by the objective function is backpropagated to tune the network parameters using the gradients of the objective function and a stochastic optimization algorithm. An illustration of the dataflow in neural network training is shown in Figure 3.2.

3.4.2 Inference

Once a neural model is trained for OOV classification, we can use the model to predict the OOV classes that are likely to occur in an ASR hypothesis. The sigmoid layer takes as input a the document representation and outputs a K-dimensional vector $\hat{\mathbf{y}}$ of real values between 0 and 1. Thus, each dimension in the output of the neural network $\hat{\mathbf{y}}$ can be interpreted as

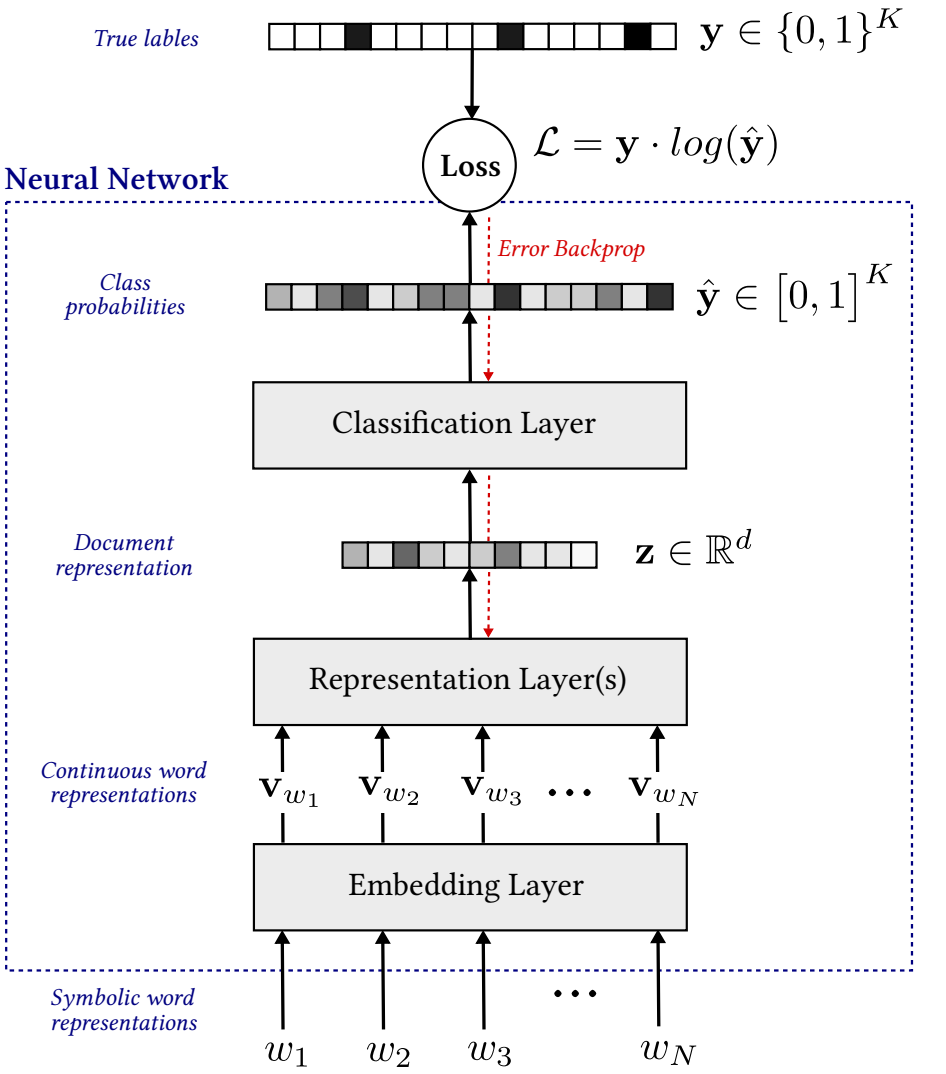

FIGURE 3.2: An illustration of the dataflow in neural network training.

the probability of a single OOV class:

$$P(C_i \mid \mathbf{z}) = \hat{\mathbf{y}}_i \tag{3.10}$$

The most probable OOV class of a spoken document corresponds to the dimension with highest probability value as follows:

$$C^* = \arg\max_C P(C \mid \mathbf{z}) \tag{3.11}$$

Since we formulate our task as multi-class multi-label classification problem, the top-Q classes that have the the highest probabilities are considered as predicted labels. The choice of Q depends on the setting. For example, when we evaluate the OOV classifiers we consider $Q = \{1, 5, 10\}$. Then, standard metrics such precision and recall are computed at these values of Q. To use the models for OOV words retrieval, the value of Q can be higher since we are interested in investigating the retrieval performance when extending the ASR lexicon

at different operating points.

3.5 OOV Words Ranking

The goal of our system is to output a set of OOV proper nouns that are highly relevant to the spoken utterance. In our setting, we build classification models that predict highly relevant OOV classes. Since each of the predicted OOV classes consists of many OOV proper nouns, we need a ranking procedure. That is, for each spoken document we need to retrieve OOV proper nouns based on their relevance to the semantic content of the document. This could be achieved by quantifying the semantic similarity between the spoken document and each candidate OOV proper noun. To this end, we use the cosine distance as a measure of semantic similarity:

$$distance(\mathbf{d}_{IV}, \mathbf{v}_{OOV}) = 1 - \frac{\mathbf{d}_{IV} \cdot \mathbf{v}_{OOV}}{\|\mathbf{d}_{IV}\|^2 \cdot \|\mathbf{v}_{OOV}\|^2} \tag{3.12}$$

where $\mathbf{d}_{IV}$ is a vector representation of a spoken document. To represent the document as a single vector, we use the averaged word embeddings as follows:

$$\mathbf{d}_{IV} = \frac{1}{|d|} \sum_{w \in d} \mathbf{v}_w \tag{3.13}$$

The ranking procedure for a single document is described as a Python function in Listing 1.

3.6 Evaluation

To evaluate the models developed in this thesis, we need to evaluate the two steps in our approach. First, the OOV classification methods are evaluated using standard metrics in multi-class multi-label classification. Next, the OOV words retrieval performance of our system is evaluated using standard metrics in information retrieval. It is worth pointing out that we adopt an intrinsic evaluation for our models. That is, we evaluate the performance of our system in retrieving target OOV proper nouns from in-vocabulary contexts. Based on the results of this intrinsic evaluation, the best performing model will be used to update the lexicon of the ASR so we can investigate if our approach actually improves the performance of the speech recognizer.

3.6.1 OOV Classification Evaluation

To evaluate the classification performance, we use an evaluation setting similar to that proposed in (Weston, Chopra, and Adams, 2014). The label space for multi-class multi-label classification can be relatively large, which means each document has few relevant labels. As it will be described in Chapter 4, we build classifiers to map documents into one or more labels out of 1000 possible labels. To address this issue, the probabilities of the labels (the

```python
def rank_OOVs(doc_embedding, oov_embeddings, oov_classes, class2oov):
    """
    Arguments:
        doc_embedding: document vector (numpy array)
        oov_embeddings: OOV word embeddings (dictionary of numpy arrays)
        oov_classes: predicted OOV classes (list)
        class2oov: class to OOV words mapping (dictionary)
    Output:
        A ranked list of OOV words based on cosine distance (list)
    """
    ranked_OOVs = list()

    for c in oov_classes:
        # get the OOVs within class c
        candidate_OOVs = [w for w in class2oov[c]]

        # rank OOVs in class c based on similarity with doc embedding
        word2score = {}

        for w in candidate_OOVs:
            w_embedding = oov_embeddings.get(w)

            # compute cosine distance
            word2score[w] = cosine_distance(doc_embedding, w_embedding)

        # sort OOV words by distance score
        ranked = sorted(word2score.items(), key=lambda x: x[1])
        ranked_OOVs.extend(w for w, d in ranked)

    return ranked_OOVs
```

LISTING 1: Python code for the ranking procedure for a single document.

output of the neural network) can be used to rank all labels from the most probable to the least probable. Then, rank-based evaluation metrics are used to assess the quality of the predicted labels in the top portion of the ranked label list. These metrics are precision at top-Q predicted labels ($P@Q$) and recall at top-Q predicted labels ($R@Q$). We compute $P@Q$ and $R@Q$ for a single test document as follows:

$$P@Q = \frac{\text{Number of correct labels in top-}Q}{Q} \tag{3.14}$$

$$R@Q = \frac{\text{Number of correct labels in top-}Q}{\text{Number of ground-truth labels}} \tag{3.15}$$

we use $P@1$, $R@5$, and $R@10$ as our evaluation metrics for our OOV classification methods. These metric are calculated for each evaluation document then the they averaged over all documents in the evaluation set (i.e., macro-average metrics).

3.6.2 OOV Words Retrieval Evaluation

The essential goal of our system is to retrieve a set of OOV proper nouns $\tilde{\mathcal{V}} = \{v_{OOV_1}, \ldots, v_{OOV_M}\}$ that are likely relevant to the spoken document. Then, words in $\tilde{\mathcal{V}}$ are added to the lexicon before running second-pass decoding with the extended lexicon. To evaluate the performance of our system, we use rank-based metrics at different operating points (that is, different values of M). In particular, methods that have better recall are preferred. In our setting, we do not focus much on the precision of OOV word retrieval, since the ASR lexicon will be extended with relevant OOV words but only a few of those actually occur in the reference transcription. We compute the metric recall at top-M ($R@M$) for OOV word retrieval for each document as follows:

$$R@M = \frac{\text{Number of target OOV PNs in top-}M}{\text{Number of target OOV PNs}} \tag{3.16}$$

here, the target OOV PNs are the OOV proper nouns that actually occur in the ground-truth transcription. This formulation of recall is the standard for evaluation information retrieval tasks. To compute the recall of an evaluation dataset, we use the macro-average recall. In addition, we use the mean of the ranks of the top-ranked target OOV proper noun in each evaluation document as an evaluation metric. We refer to this metric as *mean rank*. Given an evaluation dataset E, *mean rank* is computed as follows:

$$\text{mean rank} = \frac{1}{|E|} \sum_{d \in E} \text{rank}_{top} \tag{3.17}$$

where $rank_{top}$ is the rank of the target OOV that has the highest rank in the OOV proper noun list $\tilde{\mathcal{V}}$.

Chapter 4

Experimental Setup

In this chapter, we present an overview of the experimental methodology used in this thesis. This chapter includes information about the data used for training and evaluating our models, as well as the choice of the hyperparameters of the neural architectures.

4.1 Data

The goal of this thesis is to explore methods for OOV proper nouns recovery for large vocabulary speech recognition systems in the French language. Therefore, we train our models using textual data from public French Wikipedia articles and evaluate our models with French broadcast from Euronews channel. This section describes each of these corpora and presents some summary statistics about them.

4.1.1 Training Corpus

A training corpus is necessary to train context models for OOV retrieval, obtain an inventory of OOV proper nouns, and build word embeddings for both in-vocabulary and OOV words. The training corpus should be large to provide good coverage over OOV proper nouns and their in-vocabulary word contexts. Therefore, we choose to train our models using public data from French Wikipedia articles.

The Wikipedia data used in this thesis has been extracted and preprocessed within the course of another project. More precisely, the OOV proper nouns in the corpus were automatically identified using TreeTagger (Schmid, 1995). In addition, function words are removed and each word token is replaced by its lemma in the TreeTagger's output. The final preprocessed corpus consists of approximately 1.4 million French articles. Table 4.1 shows some statistics about the training corpus. In Table 4.1, # *unique words* is the vocab size of the corpus, while # *unique OOV PNs* is to the number of unique OOV proper nouns in the corpus.

4.1.2 Development Corpus

We use a held-out of 1% of the training data ($\sim$14000 documents) as a development set to tune the hyperparameters of the neural models developed in this thesis. The development corpus has the same characteristics as the training corpus.

TABLE 4.1: Statistics about the Wikipedia training corpus.

	whole corpus	*average per article*
# sentences	26.6M	18.9
# word tokens	268.0M	190.8
# OOV PN tokens	20.8M	14.77
# unique words	1.83M	—
# unique OOV PNs	1.75M	—

4.1.3 Test Corpus

We use the test corpus of Sheikh et al. (2017) to evaluate our methods in this study. This corpus comes from the website of the broadcast channel *Euronews*. The corpus consists of 2048 French textual news reports (from January 2014 to June 2014). We use this corpus to evaluate the performance of classification models as well as the OOV proper noun retrieval methods developed in this thesis. Table 4.2 presents statistics about this corpus.

TABLE 4.2: Statistics about the Euronews test corpus.

	whole corpus	*average per article*
# sentences	24,040	12.3
# word tokens	510,351	260.1
# OOV PN tokens	13,768	7.0

4.2 Word Representations

Continuous word representations, or word embeddings, are exploited in this thesis for three purposes: (1) to initialise the embedding layer of the neural models, (2) to build vector representations for OOV words, (3) to build document representations by taking the average of the word embeddings in the document. Representing OOV words and documents in a common space enables us to perform ranking using cosine similarity.

We use the skip-gram model of `word2vec` to build word representations. We choose this model over the continuous bag-of-words model (CBOW) for two reasons:

- the analysis of Mikolov et al. (2013) has shown that skip-gram model outperforms CBOW for semantic tasks. Given the nature of our task, we are interested in word representations where the topical properties of the word are captured, not its syntactic properties.
- For the OOV proper noun retrieval task, the investigation of Sheikh et al. (2017) has shown that representations built by the skip-gram model yield better retrieval performance.

Furthermore, there are many hyperparameters to tune for the skip-gram model. The most crucial parameters are the dimensionality of the vectors and the size of the context window. We use the same settings as in Sheikh et al. (2017). The authors have conducted a systematic

grid search over many possible configurations and reported that vectors of 400 dimension, context window of 20 words, and hierarchical softmax for training the embeddings yielded optimal performance.

We consider only the words that occur at least four times in the training corpus. This setting gives 630k OOV PN words (out of the 1.75 million OOV proper nouns that were identified using TreeTagger).

4.3 OOV Word Clusters

The settings given in section 4.2 yielded a an inventory of 630k OOV proper nouns each with a 400-dimensional embedding. We use K-means to obtain cluster from the embeddings and consider these clusters as classes to predict in the OOV class prediction step. For K-means clustering, there are two parameters to set; the number of clusters (K) and the maximum number of iterations. We set the number of clusters to 1000 and perform K-means clustering for 10 iterations.

Even though there are many choices for the number of cluster K, the reasons of choosing to represent our OOV classification output space in 1000 classes is related to the size of the vocabulary of our ASR system. The lexicon of our ASR system consists of 96,712 words. Since we aim to perform document-specific extension of the lexicon with relevant OOV proper nouns, the list of relevant OOV proper nouns should be smaller in size compared to the size of the lexicon. This is necessary in order not to increase the complexity of the ASR search space. Given that our OOV proper nouns pool consists of approximately 630k words, grouping words in this inventory into 1000 clusters would result in 630 words on average in each cluster (more information about the output of the clustering algorithm is provided in chapter 5). Therefore, if the OOV classifier makes an incorrect prediction, around 630 OOV proper nouns would be added to the lexicon. If the number of K was smaller, each incorrect prediction the classifier makes would result in more irrelevant OOVs. Large values of K would result in increasing the output space of the classification task and makes the prediction more computationally complex. We argue that the value we have chosen is reasonable given the problem at hand, although conducting a systematic investigation of other options regarding the number of clusters (K) is encouraged for future work.

4.4 Experimental Setup for OOV Classification

In this section, we give information about the design choices and the hyperparameters in the experiments.

Baseline

We use the standard text classification tool `fasttext` (Joulin et al., 2016) as a baseline for our classification experiments. A `fasttext` classifier is a feed-forward network with one hidden layer and softmax output layer that represents a probability distribution over the output

space. To make surethe results of `fasttext` models comparable to our models, we set the dimensionality of the word embeddings to 400, and context window size to 20 words. Other settings are kept as the defaults. The word embeddings for `fasttext` models are learned from scratch.

Deep Averaging Networks

For deep averaging networks DANs, we conduct experiments with the following settings:

- Number of hidden layers in the network. We evaluate DANs with 1, 2, and 3 hidden layers.
- Number of units in each layer in the network (or width of each layer). We evaluate DANs with 200, 400, 800, 1200, and 1600 units in each hidden layer.
- For regularization, we use dropout (Srivastava et al., 2014) and batch normalization. Our preliminary experiments showed that dropout rate of 0.2 is optimal.
- We use static word embeddings for DAN models. That is, the embedding layer in the network is frozen and parameters are retrained.

Convolutional Neural Networks

Convolutional Neural Networks have many hypermeters to tune. In this thesis, we do not aim to attempt all possible configurations, but we experiment with the following aspects:

- Number of filters in the convolutional layer. We experiment with 50, 100, 200, 300, 600, 1000, and 2000 filters.
- The size of the window of the filter. We experiment with 3, 4, 5, 6, 7, 8, and 9 window sizes.
- We use one convolutional layer followed by a max pooling layer.
- For regularisation, we use dropout with 0.2 dropout rate.
- Our preliminary experiments showed that trainable word embeddings slightly improves the performance of convolutional models. Thus, we set the embedding layer to be trainable in all experiments that involve convolutional models.

Recurrent Neural Networks

Given the size of our training data, exploring the hyperparameter space of RNNs is not feasible. Therefore, we keep the experimentation minimal. We use the following settings:

- We use a bidirectional Gated Recurrent Unit (GRU).
- We conduct two experiments to investigate the impact of the hidden state size (400 and 800).
- For regularisation, we use dropout with 0.2 dropout rate.
- To speed up the training, we use static word embeddings for recurrent models.

Furthermore, we use Adam optimiser (Kingma and Ba, 2014) and early stopping for all experiments. It is worth pointing out that the embedding layer in our experiments are initialised with the word embeddings that we have already trained (section 4.2).

Chapter 5

Experimental Results

In this chapter, the results of the experiments conducted in this thesis are reported and analyzed. We first describe the data and present some statistics about OOV proper nouns and OOV class distributions. Then, we show some examples of the OOV classes we obtained by applying *K*-means clustering on the word embeddings. In addition, we show the results of our classification experiments using the different neural models we have trained. Finally, we report OOV words retrieval performance of our system given on the test data.

5.1 Data Analysis

We used French Wikipedia articles as described in section 4.1.1 to train our models. However, we observed that some articles are very long and decided to set the maximum length of a document to 500 (preprocessed) word tokens. If a Wikipedia article is larger than 500 words, the article is partitioned into equally sized documents. For example, if an article consists of 650 words, two segments of length 325 words are made. Likewise, if an article is made up of 1200 words, three documents of size 400 will be produced. We also observed that some articles are very short in length, these are probably empty articles that only contain section titles without content. We decided to exclude short articles by setting the minimum length for a training document to 10 words. Our final training dataset constitutes of 1,475,345 documents. We held 1% of this set to be our development set. The average document length after preprocessing is 140.6 words (without OOVs). Figure 5.1 shows the distribution of documents in our training dataset by document length. We can observe that the majority of the documents contain less than 100 words.

We also studied the distribution of OOV proper nouns in both training and test corpus. Table 5.1 shows summary statistics about the distribution of OOV proper nouns. As expected, one can observe a difference in OOVs distribution between the training and test corpus due to the different domains they represent. Wikipedia articles are more diverse in content than Euronews reports which usually cover a single event. A document in our training data has 14.08 OOV proper nouns on average compared to 4.11 OOV proper nouns in the test data.

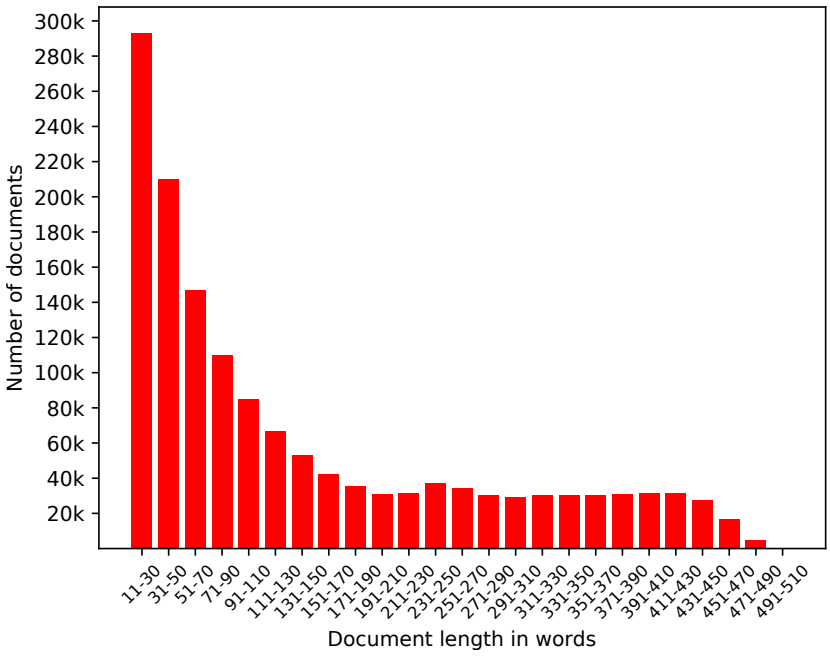

FIGURE 5.1: Distribution of documents in our training data by document length (in words).

5.2 OOV Word Classes

As described in chapter 3 and 4, *K*-means clustering was used in order to group OOV proper nouns into classes. The motivation to create these classes is to scale down the output space into OOV classes instead of OOV word types. Given that our word embeddings were learned using the skip-gram model with a large context window, the embeddings are expected to encode semantic information about the words. That is, words that occur in similar topical contexts should be nearby in space. Thus, clustering words by their semantic vectors should yield OOV proper noun clusters that are topically coherent.

We are not aware of any method that quantitatively evaluates the quality of the clusters produced by *K*-means. Therefore, we conducted a qualitative analysis of the created clusters

TABLE 5.1: Summary statistics of OOV proper noun distribution in documents in training and test corpus.

	Training corpus (Wikipedia)	Test corpus (Euronews)
Mean	14.08	4.11
Std. dev.	15.90	4.23
Minimum	1	1
Maximum	451	49
25th perc.	4.0	2.0
Median	9.0	3.0
75th perc.	18.0	5.0

TABLE 5.2: Some example clusters.

cluster 1	**cluster 2**	**cluster 3**	**cluster 4**	**cluster 5**
taoïste	bengali	tretiakov	muscat	d4
shan	malayalam	ossip	riesling	d5
taoïsme	télougou	benois	carignan	e4
calligraphe	tagore	kasimir	sarment	cf6
confucianisme	pune	larionov	gewurztraminer	e5
zhi	parsi	orloff	chasselas	a5
shou	dravidien	serov	gamay	e6
taizong	kannada	grabar	vitis	nimzo-indienne
confucianiste	marathi	roerich	effervescent	cf3
huangdi	gujarati	levitan	grenache	d6
nei	cingalais	melnikov	syrah	g3
wade-giles	bengal	répine	chaptalisation	b5
guang	padma	roublev	hl	cc3
chiji	kolkata	bakst	malvasia	b3
yangzhou	rabindranath	chtchoukine	crémant	g6
zuo	dhaka	vasnetsov	argilo-calcaires	b4
huizong	pandit	gorky	vinifera	fe7
	brahmi	artel	jurançon	e3
	varanasi		moscato	c4

to check how coherent they are. Some examples of the created clusters are shown in Table 5.2. One can observe that the clusters exhibit some coherence. We find cluster 5 interesting because it was difficult to identify what words such 'c4' and 'd6' mean, but 'nimzo-indienne' is a an defensive opening technique in chess. So, we can see that the other words in the cluster are positions on the chess board.

Furthermore, we investigated the distribution of the OOV classes in the training and test data. Summary statistics are displayed in Table 5.3.

TABLE 5.3: Summary statistics of OOV class distribution in documents in training and test corpus.

	Training corpus (Wikipedia)	Test corpus (Euronews)
Mean	5.28	1.99
Std. dev.	4.81	1.60
Minimum	1	1
Maximum	172	15
25th perc.	2.0	1.0
Median	4.0	1.0
75th perc.	7.0	2.0

5.3 OOV Classification Experiments

In this section, we report the OOV classification results on the development set (Wikipedia) and test set (Euronews). Information about the data used for evaluation is given in chapter

TABLE 5.4: Deep averaging networks classification results on development set (Wikipedia) and test set (Euronews).

DAN Layout (depth × width)	Dev set (Wikipedia)			Test set (Euronews)		
	P@1 (%)	R@5 (%)	R@10 (%)	P@1 (%)	R@5 (%)	R@10 (%)
`FastText` (1 x 400)	67.63	44.11	53.22	31.48	38.86	46.95
1 x 200	68.22	44.81	54.27	34.91	42.85	49.87
1 x 400	69.96	44.88	54.31	35.84	42.98	50.36
1 x 800	68.97	45.00	54.56	35.51	42.32	49.80
1 x 1200	68.87	44.98	54.49	36.71	41.92	49.68
1 x 1600	68.68	44.98	54.52	35.13	42.78	49.95
2 x 200	70.09	45.42	54.94	38.67	45.20	53.77
2 x 400	71.10	46.24	56.07	39.32	45.70	53.77
2 x 800	71.97	46.85	56.54	41.34	47.11	54.08
2 x 1200	72.20	47.24	57.00	40.41	46.83	54.07
2 x 1600	72.42	47.33	57.27	40.41	**47.12**	**54.56**
3 x 200	67.23	67.23	53.34	36.38	43.41	51.39
3 x 400	68.58	44.65	54.40	35.78	44.11	52.04
3 x 800	71.74	46.64	56.26	40.36	46.70	53.91
3 x 1200	72.30	47.39	56.92	40.63	47.09	54.30
3 x 1600	72.73	47.56	57.24	41.39	46.74	54.10
3 x 2400	**73.24**	**47.78**	**57.49**	**41.72**	47.06	54.48

4. We report the classification performance for each of the OOV classification methods used in this thesis. By using two corpora of different domains, we aim to investigate the effect of changing the data domain on the classification performance of the neural models. Models that are robust to the domain change are desirable. The development set is a held-out data from the Wikipedia corpus; therefore this dataset is an in-domain data. On the other hand, the Euronews test corpus is out-of-domain written news reports that have different properties than Wikipedia documents.

5.3.1 Deep Averaging Networks

Table 5.4 shows the classification results for each deep averaging network (DAN) developed in this study. We are interested in evaluating how the depth of the network (i.e., the number of hidden layers) and the width of each layer (i.e., measured by the number of units in the layer) could affect the classification performance. As discussed in chapter 4, we consider the standard classifier in FASTTEXT as a baseline.

From Table 5.4, we can observe that DAN classifiers perform better on development dataset (in terms of both precision and recall) than the Euronews dataset. This observation is expected since the development dataset is a disjoint set from the Wikipedia corpus that was used for training. Thus, it is not surprising that the performance of the classifiers degrades on the Euronews test data.

Moreover, increasing the width of the hidden layer in one-hidden-layer DANs does not seem to improve performance. In fact, all one-hidden-layer DANs seem to show almost identical performance regardless of the width of the hidden layer. The two-hidden-layer DANs consistently outperform their one-hidden-layer counterparts. One can observe a consistent performance gain in terms of precision and recall on the development set, while in the Euronews test set the trend does not seem to hold especially for precision. For three-hidden-layer DANs, an improvement on performance can only be observed when increasing the width of hidden layers. For example, and even though the difference is probably not significant, the (3×400)-DAN does not perform as well as its (2×400) counterpart. We hypothesise that increasing the width in three-hidden-layer DANs improves precision and recall on both evaluation sets. To validate this hypothesis, we conducted one more experiment with a very wide (3×2400)-DAN and it outperformed all other models in the Wikipedia development set, as well as the Euronews dataset in terms of precision, while recall on the test set does not improve. The best classifier among those reported in Table 5.4 is the (3×2400)-DAN on the development set, and the (2×1600)-DAN on the test set. Thus, the (2×1600)-DAN should be preferred for our task due to its competent performance on the test set in terms of recall.

The results reported in Table 5.4 might contradict with some common believes among deep learning practitioners regarding the relationship between the depth of the neural network and its observed performance. In our experiments, increasing the depth of the neural network does not necessarily result in performance gains. While neural networks with higher capacity could perform better, the capacity should be viewed as both the depth and width of the network. We conclude that the design of deep averaging networks should take into account both the depth of the network as well as the width of each of layer in order to obtain a significant gain in performance.

5.3.2 Convolutional Neural Networks

Using convolutional neural networks (CNNs) for text classification requires careful tuning of many hyperparameters in the network. Conducting an exhaustive search over all possible configurations for CNNs is not feasible within the course of this thesis. However, we explore two important design choices; the window size and the number of filters. Moreover, we experiment with neural models that have only one convolutional layer as previous work has shown that shallow convolutional networks for text classification give comparable performance to convolutional networks (Le, Cerisara, and Denis, 2017).

In the first experiment using CNNs, we keep all hyperparameters of the model fixed and we investigate the impact of various window sizes for the convolutional filter. All models consists of one convolutional layer of 300 filters, followed by a max pooling layer, then a ReLU dense layer. The results of this experiment are shown in Table 5.5. We do not observe any trend regarding the change of the window size and the classification performance. The differences between model are not significant and incosistent in terms of precision and recall improvement. Based on the results in Table 5.5, we decided to set the window size for the convolutional layer to 3.

TABLE 5.5: The impact of the window size of the Convolutional filter on the classification results on development set (Wikipedia) and test set (Euronews).

window size	Dev set (Wikipedia)			Test set (Euronews)		
	P@1 (%)	R@5 (%)	R@10 (%)	P@1 (%)	R@5 (%)	R@10 (%)
3	**69.22**	**45.37**	54.54	37.20	**43.64**	**52.12**
4	68.91	45.12	**54.66**	36.55	42.64	50.65
5	69.06	45.02	45.02	**37.36**	42.56	50.71
6	68.27	44.79	54.11	34.97	41.81	49.38
7	68.89	44.88	54.25	32.24	42.47	50.31
8	68.74	44.77	54.07	34.69	42.13	51.48
9	68.10	44.56	53.77	30.99	41.32	49.15

In the second experiment with CNNs, all hyperparameters of the model are kept fixed, and we investigate the impact of increasing the number of filters in the convolutional layer. In our first experiments, we observed that a window size of 3 seems to give the best performance compared to other window sizes. Thus, we set the window size to 3 in all models in the second experiment. The results of the second experiment are shown in Table 5.6. It can be observed that increasing the number of convolutional filters lead to consistent performance gains on the in-domain development set, but not necessarily on the out-of-domain test set. Note that the number of filters represents the dimensionality of the document representation due to the max-pooling operation.

From Table 5.6, we can observe that a convolutional layer with 1000 filters gives the best performance on the out-of-domain test set. Thus, this model should be preferred over other models. Nevertheless, none of the convolutional models outperforms DAN models which their performance reported in the previous section. This finding is surprising given the reported performance of CNNs in the literature for multi-class multi-label text classification. It is worth pointing out that we also experimented with CNNs that have filter banks of various sizes (e.g., 3-4-5-6), CNNs with k-max pooling instead of max pooling, and CNNs with additional non-linear layer before the output layer. None of these models outperforms the models reported in Table 5.6.

5.3.3 Recurrent Neural Networks

We also conduct experiments with bidirectional Gated Recurrent Units (BI-GRU) for the classification task at hand. Since our training data is large, training many RNN-based models is not feasible because recurrent neural models are slower to train than other models. Therefore, we only report the result for two BI-GRU models: a 400-dimensional BI-GRU and a 800-dimensional BI-GRU. The dimensionality of the model here refers to the size of the hidden state. For example, the 400-dimensional BI-GRU consists of two GRUs each with 200 units hidden state, one GRU processes the input sequence in the forward direction while the other GRU processes the input sequence in the backward direction.

TABLE 5.6: The impact of the number of filters in the Convolutional layer on the classification results on development set (Wikipedia) and test set (Euronews).

number of filters	Dev set (Wikipedia)			Test set (Euronews)		
	P@1 (%)	R@5 (%)	R@10 (%)	P@1 (%)	R@5 (%)	R@10 (%)
50	58.32	40.33	49.51	27.94	39.36	47.32
100	67.62	44.84	54.21	35.24	44.05	51.57
200	68.61	45.27	54.33	36.22	43.88	51.10
300	69.22	45.37	55.54	37.20	43.20	52.12
600	70.23	46.11	55.39	36.71	44.35	52.01
1000	70.52	46.23	55.50	**38.67**	**45.69**	**53.18**
2000	**70.54**	**46.67**	**56.05**	36.06	44.03	52.56

TABLE 5.7: (BI-GRU) classification results on development set (Wikipedia) and test set (Euronews).

dimensionality	Dev set (Wikipedia)			Test set (Euronews)		
	P@1 (%)	R@5 (%)	R@10 (%)	P@1 (%)	R@5 (%)	R@10 (%)
400	71.28	46.46	56.02	40.63	**47.75**	55.31
800	**72.48**	**47.21**	**56.73**	**41.67**	47.72	**56.06**

From Table 5.7, it can be observed that the 800-dimensional BI-GRU outperforms the 400-dimensional BI-GRU. On the Euronews test set, the 800-dimensional BI-GRU shows optimal performance compared to the other models investigated in this thesis regarding precision and recall at 10. This impressive performance of BI-GRUS is not surprising given that the hidden states of gated recurrent architectures have a large capacity for encoding sequential information even when the sequence is long.

5.3.4 Discussion

In this section, we summarise and discuss the results of the classification experiments. Figure 5.2 shows a summarised overview of the performance of the classifier on the Euronews test set. The best performing classifier in each category is represented in this figure.

One of the observations that we find surprising is the poor performance of the convolutional models compared to the other models developed in this study. We expected the convolutional models to outperform the DAN models due to their ability to encode the order of the words in the input sequence. However, in our experiments, DAN models outperform convolutional models by a significant margin. Therefore, we conclude that CNNS models with one convolutional layer do not perform well on the multi-class multi-label classification of long documents. We wonder whether or not adding more convolutional layers to the network architecture would lead to any performance gains. Therefore, it would be

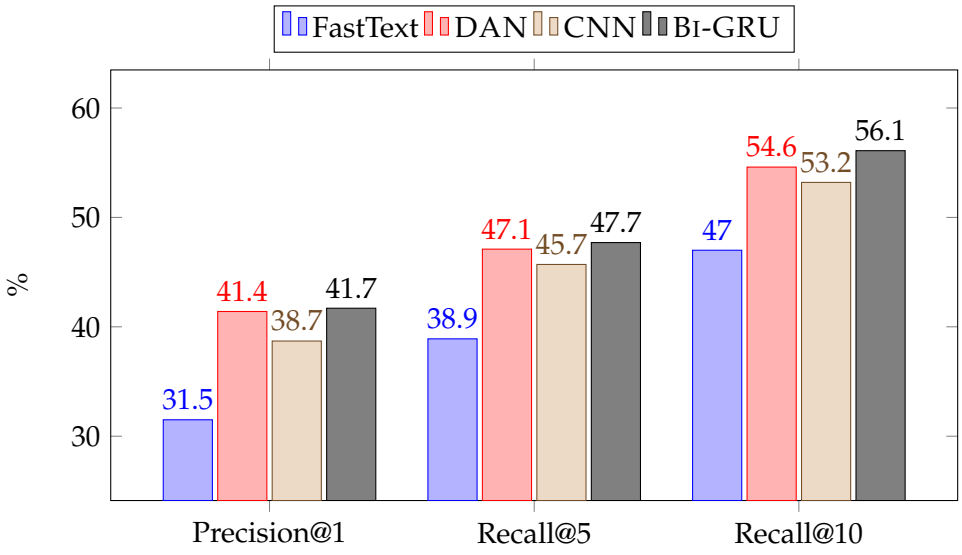

FIGURE 5.2: Classification performance measured by Precision@1, Recall@5, and Recall@10 metrics on Euronews data (out-of-domain test set).

interesting for a future word based on this study to investigate the performance of deep convolutional networks on the classification task.

Although the Bi-GRUs models outperform DAN models in our experiments, DAN models give an impressive performance if we consider their simplicity. Therefore, when computational resources for training deep neural networks may not be available, simple DAN models make a competent alternative. Since training and evaluating DAN models do not take a long time, it would be feasible to evaluate different possible design choices.

5.4 OOV Word Retrieval Performance

In order to evaluate the performance of our document-specific ASR lexicon extension methods for the OOV proper nouns retrieval task, we conduct two experiments: (1) In Experiment I, the goal is to investigate the retrieval performance of OOV proper nouns given their in-vocabulary contexts from textual Euronews documents, (2) In Experiment II, the goal is to investigate the retrieval performance of OOV proper nouns given the output of the ASR, which is a sequence of in-vocabulary words but with some noise due to the errors in the ASR hypothesis.

For Experiment I, the OOV proper nouns retrieval performance for different methods is shown in Figure 5.3. In this experiment, the test data set described in chapter 4 is used. In Figure 5.3, the x-axis represents the number of OOV proper nouns (in logarithmic scale) that could be added to the ASR lexicon before running second-pass decoding. The y-axis represents the OOV proper noun retrieval performance measured by the average recall on the Euronews test set. For example, if we decide to extend the ASR vocabulary by adding top 4K words in the ranked list of OOV proper nouns, we observe the corresponding recall at the y-axis. From Figure 5.3, we can observe that the methods we developed in this thesis perform better than a static extension of the ASR lexicon based on OOV frequency statistics.

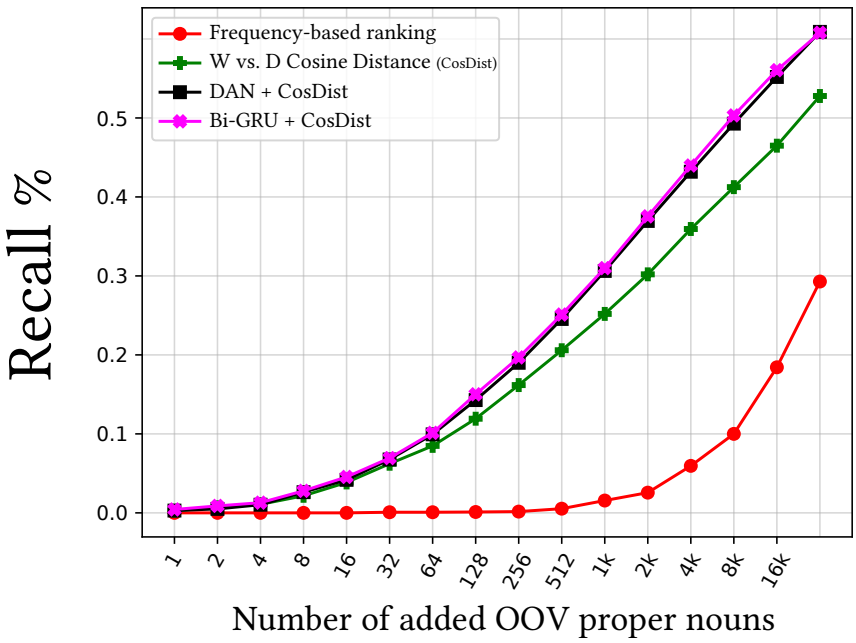

FIGURE 5.3: OOV Word Retrieval performance measured by the recall for Experiment I.

Our methods aim to extend the ASR lexicon in a dynamic approach based on each spoken utterance and its in-vocabulary context.

For our document-specific methods for ASR lexicon extension, we can observe that incorporating predictions from OOV classifiers improves the retrieval performance. Our top performing classifiers —the (2×1600) DAN and the 800 BI-GRU— were utilized to improve the ranking of OOV proper nouns. Both methods that incorporate the classifiers outperform the direct ranking approach using the averaged word embeddings for each spoken document.

To get a closer look at the performance of our document-specific methods for ASR lexicon extension, we investigate the performance when 4K and 8K OOV proper nouns are added to the lexicon (~5% and ~10% of the size of our ASR lexicon, respectively). The results of this investigation are shown in Table 5.8. Even though the difference in performance may not be clear in Figure 5.3, information in Table 5.8 highlights the difference. We can observe that our OOV classification-based ranking improves recall and mean rank by a considerable margin. By relying on predictions from the classifiers, up to 7% improvement on recall can be obtained when extending the ASR lexicon by ~5% of its original size. Our top performing classifier —800 BI-GRU– gives best results in terms of recall (43.98%) and mean rank (3134). The mean rank metric could be interpreted as follows: in a ranked list of OOV candidates for each test document, mean rank is the average rank of the first relevant OOV proper noun in the list. These results are very promising, and they indicate that improving the performance of the OOV class prediction would definitely improve the OOV proper noun retrieval.

In Experiment II, we use an evaluation set that has not been used for tuning the models.

TABLE 5.8: Performance investigation of our document-specific approaches for extending the ASR lexicon for Experiment I. W vs D Cosine Distant refers to the cosine distance between OOV embedding and document embedding.

Ranking method	Document-specific?	Classifier-based?	recall@4K	recall@8K	Mean rank
Frequency-based ranking	No	No	5.95%	10.00%	—
W vs. D Cosine Distance (CosDist)	Yes	No	35.94%	41.24%	55769
DAN + CosDist	Yes	Yes	43.19%	49.28%	3273
Bi-GRU + CosDist	Yes	Yes	**43.98%**	**50.32%**	**3134**

This dataset consists of 296 video news reports with their reference transcription. The OOV proper nouns in the reference transcriptions were extracted, and we are interested in evaluating the performance of our system when the output of the ASR is given, as the typical usage of the system in practice. The results of this evaluation are depicted in Figure 5.4.

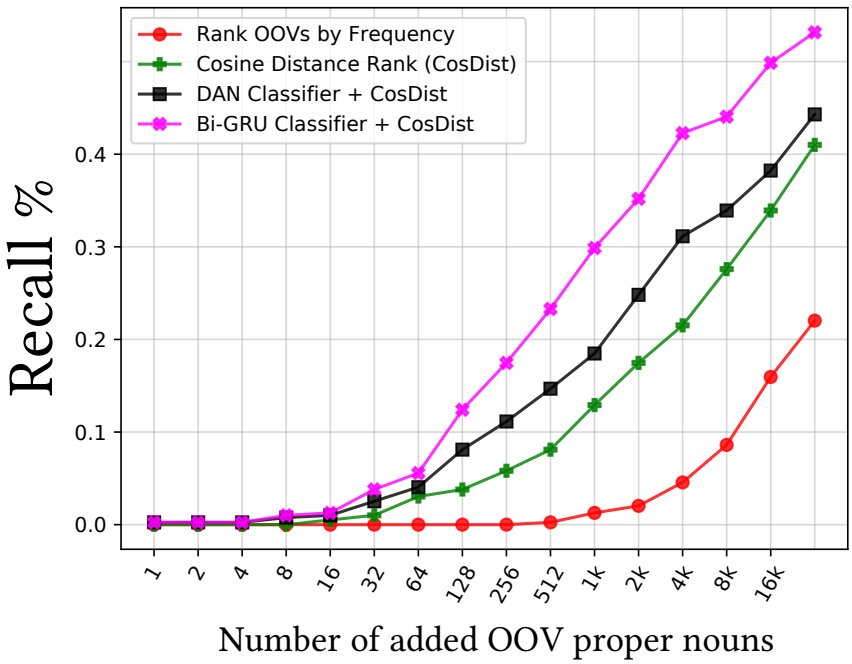

FIGURE 5.4: OOV Word Retrieval performance measured by the recall for Experiment II.

In Figure 5.4, the most interesting observation is that the performance of our document-specific extension approaches degrade, and the only model that does not seem to be affected when the ASR hypothesis is used as input is the model that is based on the BI-GRU classifier. We hypothesise that the noise in the ASR transcriptions has an adverse impact on the performance of the averaging models, which take the embedding of each word in the ASR hypothesis into account. Conversely, the reset and update gates in the BI-GRU classifier seem to make the classifier more robust to the noise in the input since this gating mechanism could enable the model to be more adaptive to relevant inputs. To highlight the difference

between the models, Table 5.9 presents the recall scores for the different methods at 4K and 8K.

TABLE 5.9: Performance investigation of our document-specific approaches for extending the ASR lexicon Experiment II. W vs D Cosine Distant refers to the distance between OOV embedding and document embedding measured by cosine similarity.

Ranking method	Document-specific?	Classifier-based?	recall@4K	recall@8K	Mean rank
Frequency-based ranking	No	No	4.56%	8.61%	—
W vs. D Cosine Distance (CosDist)	Yes	No	21.52%	31.14%	38228
DAN + CosDist	Yes	Yes	31.14%	33.92%	5390
Bi-GRU + CosDist	Yes	Yes	**42.28%**	**44.05%**	**1848**

5.5 Recognition of OOV Proper Nouns

In this section, we investigate the impact of our document-specific lexicon extension approach on the speech recognition performance. A list of relevant OOV proper nouns is retrieved by the best performing model developed in this study (BI-GRU). We extract top 4K OOV proper nouns from the for each spoken document. We construct the extended lexicon (base lexicon and 4K OOV proper nouns) per document and update the language model per document. We perform a speech recognition with the updated ASR system. Table 5.10 word error rate (WER) and proper name error rate (PNER) results on test speech corpus using the extended lexicon. PNER is obtained by the reference and hypothesised word-level transcriptions calculating substitution, deletion, insertion errors, and on the proper noun terms. From Table 5.10, we can see that adding the new OOV proper nouns to the lexicon and language model did not have a negative impact on the WER. Instead, the WER showed an improvement. This improvement is due to the recognition of OOV proper nouns and the reduction of insertion and deletion errors. The PNER reduction is larger compared to WER (6% relative improvement). This PNER reduction indicates that the relevant proper nouns were correctly transcribed by the ASR in the second-pass recognition. The results in this section were given to us by Dominique Fohr.

TABLE 5.10: WER and PNER performance of OOV proper nouns retrieval for Euronews speech test set after second-pass recognition.

System	WER (%)	PNER (%)
ASR with base lexicon	20.3	39.9
ASR with extended lexicon	**19.9**	**37.4**

5.6 Summary

This chapter presented the results of the experiments we conducted in this thesis. In this section, we summarise the results and highlight some interesting findings. We also present our attempts to explain the behaviour of the models developed in this study.

In this thesis, we propose to address the problem of OOV proper nouns recovery with a two-step approach. In the first step, the OOV classes that are likely to occur in the in-vocabulary context are predicted. In the second step, the OOV proper nouns in the predicted classes are ranked based on their relevance to the spoken context. The first step was tackled as a multi-class multi-label text classification task. In this chapter, we conducted a systematic evaluation of the performance of three neural text classification methods. Despite the reported success of shallow convolutional models for text classification tasks in the literature, we found that simple deep averaging networks –with little fine tuning– outperform convolutional models developed in this study. We are not sure whether this poor performance of convolutional models in our task is due to the large output space or the design choices for convolutional models we made in this study. However, our study gives evidence that using the default design choices for convolutional modes are not optimal for multi-class multi-label tasks if the label space is large. Because deep averaging networks are fast to train and do not require much fine-tuning, they should be preferred over convolutional models. Our top performing classifier is a bidirectional GRU that was trained for four days without much fine-tuning. This finding indicates that attention-based recurrent structures with careful tuning for this task could improve the classification performance; thus we recommend such architectures for future work. Nevertheless, the comparable performance of deep averaging networks in our evaluation suggests that these models should be considered as strong baselines for new tasks due to their design simplicity and computational efficiency.

In the second step, we use OOV class predictions to scale down the output space of the OOV proper nouns. That is, only OOV words that are members of the predicted OOV classes are considered for ranking. For OOV word ranking, we used the cosine distance between the averaged embeddings of the document and the embedding of each candidate OOV proper noun as a measure of semantic similarity. Then, OOV proper nouns are ranked based on the similarity as measured by the cosine distance. We evaluated the performance of our methods compared to an OOV frequency-based lexicon extension approach. Our document-specific methods improve the recall by a considerable margin. This shows the importance of dynamic approaches that are based on contextual information to adapt ASR lexicon for each spoken test document. Furthermore, incorporating OOV class prediction from our developed classifiers improves the performance even further. Even though the predictions made by the classifiers usually contains incorrect predictions, relying on these predictions improves the recall at different operating points. We were able to report a recall score of 42.28% (when adding 4K OOV words to the lexicon) when using OOV class predictions from the bidirectional GRU to improve the ranking. Our system could be further improved if the errors made by the classifier are reduced.

Finally, an extrinsic evaluation of the system developed in this research was conducted. Using our best performing model to extend the ASR lexicon per each test document, we have shown that the word error rate (WER) has dropped from 20.3% to 19.9%. This finding indicates that dynamic ASR lexicon extension approaches can reduce the recognition errors that are caused by the presence of OOV words.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we proposed methods for document-specific extension of ASR lexicon to recover missing OOV proper nouns in ASR transcriptions. Our approach infers the semantic content of each spoken document from its initial ASR hypothesis in order to retrieve highly relevant OOV proper nouns. To model the statistical relationship between OOV words and their in-vocabulary word contexts, we leveraged a large text corpus collected from Wikipedia articles.

Using Wikipedia as a text corpus, we were able to obtain an inventory of hundreds of thousands of OOV proper nouns. Compared to the size of our ASR base vocabulary ($\sim$ 97k words), retrieving a list of content-relevant proper nouns from this sizeable OOV inventory is not trivial. Therefore, we addressed the complexity associated with the output space with a two-step approach. In the first step, the goal was to predict the OOV classes that are likely to occur in a given context. To this end, we used K-means clustering to group OOV words into 1000 clusters where semantically similar words are members of the same cluster. Then, each cluster is considered a target class of the in-vocabulary context, and we tackle the problem of predicting the OOV class as multi-class multi-label text classification. We used state-of-the-art neural text classification models to predict OOV classes that are likely relevant to the in-vocabulary context.

In the second step, the goal was to rank the OOV proper nouns in the predicted OOV classes (n the first step), so relevant proper nouns are highly ranked in the retrieved list. To solve this problem, we proposed a ranking procedure that is based on the cosine distance between the vector representation of each candidate OOV word and the vector representation of the spoken document. Then, the cosine distance was used as a score to rank the OOV words from the most relevant to the least relevant.

The OOV word retrieval performance of our system was evaluated using rank-based metrics. We investigated the performance of our system at different operating points. That is, the impact of changing the size of the candidate OOV list on the lexicon coverage was investigated. The results of our experiments showed that our dynamic document-specific methods for ASR lexicon extension outperform static frequency-based methods by a significant margin. Furthermore, incorporating the predictions of our OOV classification models has further improved the OOV word retrieval performance. When extending the lexicon

with nearly 5% of its original size, our top performing model was able to retrieve 42% (measured by macro-average recall) of the target OOV proper nouns in the reference transcriptions.

6.2 Future Work

The results of our experiments showed the methods developed in this thesis improve the coverage of ASR lexicons in order to recognise content-relevant proper nouns correctly. In this section, we present a few research directions that could be adopted to improve the work in this thesis further. Furthermore, a few ideas on how to address the OOV words problem in the long term are described.

6.2.1 Short-term improvements

The work presented in this thesis relies on continuous word representations. The word representations in this thesis were built using the skip-gram model of Mikolov et al. (2013). However, the quality of these representations is questionable for infrequent OOV words. To build more robust representations, one can use representation models that enrich the representations with subwords units and character n-grams information (Bojanowski et al., 2016). Furthermore, the work of Singh et al. (2016) proposed to exploit orthographic similarity between words to induce representations for rare words. Building OOV word clusters using word representations induced from different representation models can also deliver insights on the topical coherence of those clusters. Thus the quality of the OOV clusters can be assessed and improved.

For the OOV classification, our experiments indicate that the OOV word retrieval performance is related to the accuracy of predictions made in the OOV classification step. Our best performing models were based on averaging feedforward neural networks and bidirectional recurrent neural networks. For the averaging model, the model could be further improved by using word dropout (Iyyer et al., 2015) which behaves as both a regularisation and data augmentation technique. For the recurrent model, attention-based recurrent structures for text classification can be explored to enhance the OOV class prediction performance (Yang et al., 2016; Liu et al., 2017).

6.2.2 Long-term improvements

Many directions could be explored to improve the work in this thesis in the long term. We describe two ideas that we consider fruitful.

In this thesis, the leverage a text corpus collected from Wikipedia articles to capture the relationship between OOV proper nouns and their in-vocabulary word contexts. However, Wikipedia has rich, structured information about the written content that we do not exploit in this thesis. Each article in Wikipedia is often associated with a set of categories. The predefined categories in Wikipedia can be considered as ground-truth labels for the supervised classification task. In addition, the statistical correlation between the categories and OOV

words can be captured. Relying on the article categories in Wikipedia would eliminate the need for OOV word clustering since each category is associated with a set of relevant OOV words by default.

Moreover, the work in this thesis relied on the semantic information in the ASR hypothesis to recover OOV words but did not consider the acoustic information in the spoken utterance. The ranking procedure can be based on spoken term discovery mechanism. That is, one can exploit the acoustic information in the speech signal to train a binary classifier to predict whether or not each OOV in the candidate list has actually in the utterance. To obtain training data for this classifier, text synthesis technology can be used to generate speech recordings of written documents automatically.

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473*.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166.

Bojanowski, Piotr et al. (2016). "Enriching word vectors with subword information". In: *arXiv preprint arXiv:1607.04606*.

Cho, Kyunghyun et al. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078*.

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

Currey, Anna, Irina Illina, and Dominique Fohr (2016). "Dynamic adjustment of language models for automatic speech recognition using word similarity". In: *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, pp. 426–432.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pp. 6645–6649.

Harris, Zellig S (1954). "Distributional structure". In: *Word* 10.2-3, pp. 146–162.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

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

Iyyer, Mohit et al. (2015). "Deep unordered composition rivals syntactic methods for text classification". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Vol. 1, pp. 1681–1691.

Jelinek, Frederick, Lalit Bahl, and Robert Mercer (1975). "Design of a linguistic statistical decoder for the recognition of continuous speech". In: *IEEE Transactions on Information Theory* 21.3, pp. 250–256.

Joulin, Armand et al. (2016). "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759*.

Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever (2015). "An empirical exploration of recurrent network architectures". In: *International Conference on Machine Learning*, pp. 2342–2350.

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom (2014). "A convolutional neural network for modelling sentences". In: *arXiv preprint arXiv:1404.2188*.

Kenter, Tom, Alexey Borisov, and Maarten de Rijke (2016). "Siamese cbow: Optimizing word embeddings for sentence representations". In: *arXiv preprint arXiv:1606.04640*.

Kim, Yoon (2014). "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kiros, Ryan, Richard Zemel, and Ruslan R Salakhutdinov (2014). "A multiplicative model for learning distributed text-based attribute representations". In: *Advances in neural information processing systems*, pp. 2348–2356.

Klakow, Dietrich, Georg Rose, and Xavier Aubert (1999). "OOV-detection in large vocabulary system using automatically defined word-fragments as fillers". In: *Sixth European Conference on Speech Communication and Technology*.

Le, Hoa T, Christophe Cerisara, and Alexandre Denis (2017). "Do Convolutional Networks need to be Deep for Text Classification?" In: *arXiv preprint arXiv:1707.04108*.

Lecouteux, Benjamin, Georges Linares, and Benoit Favre (2009). "Combined low level and high level features for out-of-vocabulary word detection". In: *Tenth Annual Conference of the International Speech Communication Association*.

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.

Liu, Ming et al. (2017). "Leveraging linguistic resources for improving neural text classification". In: *Proceedings of the Australasian Language Technology Association Workshop 2017*, pp. 34–42.

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

Mitchell, Jeff and Mirella Lapata (2010). "Composition in distributional models of semantics". In: *Cognitive science* 34.8, pp. 1388–1429.

Mitra, Bhaskar and Nick Craswell (2017). "Neural Models for Information Retrieval". In: *arXiv preprint arXiv:1705.01509*.

Parada, Carolina et al. (2010). "A spoken term detection framework for recovering out-of-vocabulary words using the web". In: *Eleventh Annual Conference of the International Speech Communication Association*.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Schmid, Helmut (1995). "Treetagger| a language independent part-of-speech tagger". In: *Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart* 43, p. 28.

Shannon, Claude Elwood (2001). "A mathematical theory of communication". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5.1, pp. 3–55.

Sheikh, Imran et al. (2017). "Modelling semantic context of oov words in large vocabulary continuous speech recognition". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 25.3, pp. 598–610.

Singh, Mittul et al. (2016). "Sub-Word Similarity based Search for Embeddings: Inducing Rare-Word Embeddings for Word Similarity Tasks and Language Modelling". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 2061–2070.

Srivastava, Nitish et al. (2014). "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

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.

Weston, Jason, Sumit Chopra, and Keith Adams (2014). "# tagspace: Semantic embeddings from hashtags". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1822–1827.

Yang, Zichao et al. (2016). "Hierarchical attention networks for document classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489.

Zhang, Ye and Byron Wallace (2015). "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1510.03820*.