Towards Learning Language Agnostic Features for NLP in Low-resource Languages

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

by

Research

by

Allen Jojo Antony
201401124
allen.antony@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2020
International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Towards Learning Language Agnostic Features for NLP in Low-resource Languages" by Allen Jojo Antony, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Radhika Mamidi
To Papa and Amma
Acknowledgments

The work presented in this thesis would not be possible without the support of all the people who accompanied me in my journey through IIIT-H. First and foremost, I would like to thank my parents for their constant support and guidance through thick and thin. Words fail to describe their importance in my life.

I was lucky to start my IIIT-H journey with a group of incredibly talented people from school: Ashwin, Harsha, Roshan, Anjali and Saumya. I’d like to think that life is a learning curve and, in a sense, life is like a supervised learning model. I want to thank all of them for being positive examples to guide my life’s loss function to a global minimum.

I was also extremely lucky to meet and bond with brilliant students in IIIT-H. In fact, this thesis would not have been possible without Vinayak and the infinite wisdom that resides in his bald head. Suhan, thank you for supporting me by accompanying me in my green diet journey. We achieved great heights together. Living, working and celebrating with Jaipal has always carried the inexpressible comfort of feeling safe with a person, having neither to weigh thoughts nor measure words. It is this feeling that has propelled me to come up with innovative ideas, share them confidently and have trust that things will get better in the hardest of times. What started off as two boys living across the hall in the hostel, has now sprouted into an extremely deep rooted relationship that is on its way to do great things in life ahead.

I would like to give my most sincere thanks to Ashima, for taking out time after her very long and tiring hours at work and sharing very educational information almost on an everyday basis without a miss. Her shared facts and wisdom allowed me to grow as a person and be a better man than I am today.

Most importantly, I would like to thank my guide and mentor Prof. Radhika Mamidi for her guidance, without which this thesis would not have been possible. My sincere thanks to her for taking me under her wing as an honors students and then as a Master’s student. Thank you very much for everything.

My words are few, but heartfelt.
Abstract

In recent years, Natural Language Processing (NLP) has gained widespread attention in many commercial and academic applications. Neoteric advances in Machine Learning and Deep Learning, aided by the rise of large annotated datasets, is the cornerstone of many state-of-the-art NLP systems and architectures. The one caveat of these advances is the availability of large annotated datasets for a particular NLP task. Since the conception of the Internet and the Digital Age, more and more information is stored digitally. Given the global nature of the current information sharing infrastructure, most of the data generated belongs to one of three languages: English, Mandarin or Spanish.

This abundance of raw data, aids and motivates the creation of annotated NLP resources in these languages. On the other hand, the paucity of annotated data in most other languages makes it a challenging task to develop Deep Learning/Machine Learning based solutions for them. Hence there is a pressing need to pay special attention to develop novel solutions capable of performing NLP tasks in a low-resource setting.

In this thesis, we attempt to tackle this data scarcity problem by introducing a novel approach for language invariant NLP which is capable of leveraging multiple monolingual datasets for training without any form of cross-lingual supervision. The proposed approach attempts to learn language agnostic features via adversarial training on multiple resource-rich languages, which can then be leveraged for inference on a low-resource language. The robustness of the proposed approach was tested on two well-defined NLP tasks:

1. **Sentiment Analysis**: For classifying the sentiment of a given document in a low-resource language we introduce the **Language Invariant Sentiment Analyzer** (LISA) architecture which learns language invariant sentiment features that outperforms the previous state-of-the-art methods on the Multilingual Amazon Review Text Classification dataset and achieves significant performance gains over prior work on the low-resource Sentiraama corpus.

2. **Open Domain Event Detection**: In the case of Open Domain Event Detection, which is a sequence labeling task, we introduce the **Multi-Lingual Sequence Tagger** (M-LiST) architecture which attained state-of-the-art performance in three languages of the TempEval2 corpus.

A detailed analysis of our research highlights the ability of our architectures to achieve state-of-the-art performance in the presence of minimal amounts of training data for low-resource languages.
Contents

Chapter Page

1 Introduction ........................................... 1
  1.1 Motivation ........................................... 3
    1.1.1 The Problem with Annotated Corpora .......... 3
    1.1.2 The Problem with Low-Resource Languages ..... 3
    1.1.3 The Problem with Cross-Lingual Resources ... 5
  1.2 Research Questions ................................. 6
  1.3 Main Contributions ................................. 6
  1.4 Thesis Organization ............................... 7

2 Background and Related Works ...................... 8
  2.1 Background ......................................... 8
    2.1.1 Recurrent Neural Network (RNN) ............... 9
    2.1.2 Long Short Term Memory (LSTM) ............... 10
    2.1.3 Gated Recurrent Unit (GRU) .................... 12
    2.1.4 Bi-directional Long Short Term Memory (Bi-LSTM) 13
  2.2 Transfer Learning .................................. 14
    2.2.1 Transfer Learning: Formal Definition ........ 14
    2.2.2 Transfer Learning: Methodologies ............. 15
  2.3 Cross-Lingual Transfer Learning .................. 18
    2.3.1 CLTL using Machine Translation Systems ...... 18
    2.3.2 CLTL using Cross-lingual Word Embeddings .... 19
    2.3.3 CLTL using Cross-lingual Resources .......... 19
    2.3.4 CLTL without Cross-lingual Supervision ...... 20

3 Multilingual Word Representation ................. 21
  3.1 Introduction ....................................... 21
    3.1.1 Word Embeddings ................................ 22
    3.1.2 Multilingual Word Embeddings ................. 23
  3.2 FastText Embeddings ............................... 25
  3.3 MUSE: Multilingual Unsupervised and Supervised Embeddings 27
  3.4 Conclusions ....................................... 29

4 Learning Language Invariant Features for Sentiment Analysis 30
  4.1 Introduction ....................................... 30
    4.1.1 Sentiment Analysis: Benefits and Use Cases ... 31
## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.2 Different Forms of Sentiment Analysis</td>
<td>32</td>
</tr>
<tr>
<td>4.1.3 Sentiment Analysis : Low-resource Setting</td>
<td>33</td>
</tr>
<tr>
<td>4.1.4 Proposed Approach : Overview</td>
<td>33</td>
</tr>
<tr>
<td>4.2 Dataset Description</td>
<td>34</td>
</tr>
<tr>
<td>4.2.1 Multilingual Amazon Review Text Classification dataset</td>
<td>34</td>
</tr>
<tr>
<td>4.2.2 Sentiraama Dataset</td>
<td>35</td>
</tr>
<tr>
<td>4.3 Multilingual Word Representation</td>
<td>36</td>
</tr>
<tr>
<td>4.4 LISA Architecture</td>
<td>37</td>
</tr>
<tr>
<td>4.4.1 Multilingual Sequence Encoder ($H$)</td>
<td>37</td>
</tr>
<tr>
<td>4.4.2 Language Discriminator ($C_L$)</td>
<td>38</td>
</tr>
<tr>
<td>4.4.3 Sentiment Analyzer ($C_S$)</td>
<td>38</td>
</tr>
<tr>
<td>4.5 Training Set-Up</td>
<td>39</td>
</tr>
<tr>
<td>4.6 Experiments and Results</td>
<td>40</td>
</tr>
<tr>
<td>4.6.1 Results for the Amazon Dataset</td>
<td>40</td>
</tr>
<tr>
<td>4.6.2 Results for the Sentiraama Corpus</td>
<td>42</td>
</tr>
<tr>
<td>4.7 Analysis and Conclusion</td>
<td>43</td>
</tr>
<tr>
<td>5 Learning Language Invariant Features for Open Domain Event Detection</td>
<td>44</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>44</td>
</tr>
<tr>
<td>5.1.1 TimeML</td>
<td>46</td>
</tr>
<tr>
<td>5.1.2 TempEval Framework</td>
<td>48</td>
</tr>
<tr>
<td>5.1.3 Event Detection : Low-resource Setting</td>
<td>49</td>
</tr>
<tr>
<td>5.1.4 Proposed Approach: Overview</td>
<td>49</td>
</tr>
<tr>
<td>5.2 Multilingual Word Representation</td>
<td>50</td>
</tr>
<tr>
<td>5.3 M-LiST Architecture</td>
<td>50</td>
</tr>
<tr>
<td>5.3.1 Feature Encoder ($H$)</td>
<td>52</td>
</tr>
<tr>
<td>5.3.2 Language Discriminator ($C_L$)</td>
<td>52</td>
</tr>
<tr>
<td>5.3.3 Sequence Labeller ($S_L$)</td>
<td>53</td>
</tr>
<tr>
<td>5.4 Training Set-Up</td>
<td>53</td>
</tr>
<tr>
<td>5.5 Experiments and Results</td>
<td>54</td>
</tr>
<tr>
<td>5.6 Evaluation Metrics</td>
<td>56</td>
</tr>
<tr>
<td>5.7 Analysis and Conclusion</td>
<td>56</td>
</tr>
<tr>
<td>6 Conclusions and Future Work</td>
<td>58</td>
</tr>
<tr>
<td>6.1 Future Work</td>
<td>59</td>
</tr>
<tr>
<td>Bibliography</td>
<td>62</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Languages on the Internet (source: Scroll.in)</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>RNN Architecture</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>LSTM Cell</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>GRU Cell</td>
<td>12</td>
</tr>
<tr>
<td>2.4</td>
<td>Bi-directional LSTM Architecture</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Categories of Transfer Learning Strategies</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Multilingual Word Embeddings (English and German)</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>MUSE Embedding Space (source: <a href="https://github.com/facebookresearch/MUSE">https://github.com/facebookresearch/MUSE</a>)</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>The Language Invariant Sentiment Analyzer (LISA) Architecture</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>LISA : Language Discriminator</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>LISA : Sentiment Analyzer</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>The Multilingual Sequence Tagger (M-LiST) Architecture</td>
<td>51</td>
</tr>
<tr>
<td>5.2</td>
<td>M-LiST : Language Discriminator</td>
<td>52</td>
</tr>
<tr>
<td>5.3</td>
<td>M-LiST : Sequence Labeller</td>
<td>53</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>36</td>
</tr>
<tr>
<td>4.4</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>42</td>
</tr>
<tr>
<td>5.1</td>
<td>48</td>
</tr>
<tr>
<td>5.2</td>
<td>55</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

"Big data isn’t about bits, it’s about talent.”

— Douglas Merrill

Machine learning techniques have been around for decades, starting in the 1960s when pioneering research in Bayesian Statistics and Probabilistic Inference paved the way for more sophisticated architectures and methodologies. With the advent and popularization of Support Vector Machines (SVMs) and Recurrent Neural Networks (RNNs) in the 1990s, machine learning moved away from knowledge-driven to data-driven methods leading to the curation of large amounts of datasets. This explosion in the amount of data gathered, merited more complexity in machine learning algorithms to enhance its learning ability resulting in the rise of deep learning. Deep learning is a subset of machine learning techniques that teaches machines to do what comes naturally to humans: learning by example.

By the 2010s, computer software and hardware (mainly the advancements in GPU architectures) had caught up to the increasing demand of Deep Neural Networks making it feasible to train for many researches and industrial entities worldwide. Today, deep learning architectures such as Convolutional Neural Networks (CNNs), Long Short Term Memory (LSTM), Deep Neural Networks (DNNs), Deep Belief Networks (DBNs) and Recurrent Neural Networks (RNNs) have been successfully applied to various domains including Natural Language Processing, Speech Processing, Machine Translation, Sentiment Analysis, Event Detection in text, Computer Vision, Bio-informatics, among many others, capable of producing staggering results comparable to, and sometimes even exceeding, human-level performance.
Natural Language Processing is at the intersection of computer science, artificial intelligence, linguistics and statistical inference, concerned with teaching computers to understand, manipulate and generate human language. Understanding a language means interpreting words and phrases in different syntactic forms, various semantic, pragmatic and phonological concepts presented in different input formats such as text, speech or images and knowing how to extract information from these concepts in a meaningful way. NLP-based systems form the core foundation that enables a wide range of applications such as Alexa (Amazon’s voice assistant), Google’s search engine, Google Translate, Interactive Voice Response (IVR) applications used in call centers and Grammarly.

Although NLP boasts significant success in various domains, computers still struggle with comprehending many facets and nuances of language that are difficult to characterize formally. While humans can easily master a new language, NLP is widely considered a challenging problem in computer science. It is the inherent nature of human languages accounting for ambiguity and imprecise characteristics that makes NLP difficult. Computers have a hard time understanding the rules that govern the flow of information in natural languages. Some of these rules can be abstract and non-explicit. For example, when sarcasm is used to deliver a humorous remark. On the other hand, some of these rules can be more straightforward and explicit. For example, the use the character “s” to signify the plurality of nouns.

For a long time, the majority of methods used to study NLP problems employed shallow machine learning models using time consuming, hand-crafted features or rule-based systems built using language-specific knowledge and syntactic rules formulated by language experts to build heuristic-based deterministic systems that could perform primitive tasks like POS-tagging, lemmatization, morphological analysis, etc. But, as linguists had soon realized, these rule-based systems perform poorly in more complex NLP tasks like sentiment analysis. This is because, it is near impossible to write an exhaustive set of rules that capture different nuances and meanings that depend on context and non-explicit information.

However, with the rise of large annotated corpora and the recent popularity and success of word embeddings (Word2Vec, GloVe, FastText, etc), neural-based supervised deep learning approaches have achieved superior results on most NLP tasks as compared to traditional machine learning models like Logistic Regression or Support Vector Machines. These methods attempt to understand the text by looking at large amounts of labeled data-points and then converting the text into higher dimensional representations and tries to learn various patterns from these representations to solve a particular task. These models can often be trained with a single end-to-end model that can learn abstract features from natural language required by the model and do not require traditional, task-specific features that are engineered or extracted by an expert. Therefore, current methods in NLP are largely driven by statistical and deep learning methods trained on large corpora. It is important to note that, in supervised approaches, annotated data is imperative as it provides the only guidance for the model to learn.
1.1 Motivation

The one drawback to applying deep supervised learning techniques to NLP is its over-reliance on the quality and the amount of the data it is trained on. Deep networks require large amounts of carefully curated labeled data for a particular task to achieve satisfactory performance. Most NLP algorithms need sufficiently large collections of annotated text that accounts for the variation in syntax and semantics of spoken language.

1.1.1 The Problem with Annotated Corpora

Undoubtedly, the creation and maintenance of annotated corpora is a resource and labour intensive task as well as being severely time consuming. A typical annotation project usually requires careful design and development before undergoing several stages of quality assessment that evaluates inter-annotator agreement metrics. Another downside to annotated corpora in NLP is that it is usually language specific. Therefore, without cross-lingual supervision, a model trained on one language cannot be easily reused for another language. Annotated data is also task specific which means that it cannot easily be reused for another task. Moreover, annotated corpora is also usually domain specific. Hence, a model trained on a Twitter sentiment analysis corpus will perform poorly for sentiment analysis in movie review domain. In a rapidly changing field like NLP where new challenges and solutions are proposed on a seemingly daily basis, this exclusive dependence on annotated data is risky and not the most optimal approach. Therefore, alleviating the reliance on annotated data is one of the major motives behind this thesis.

1.1.2 The Problem with Low-Resource Languages

In a recent study conducted in 2006 [83], it was stated that out of approximately 7000 major language groups spoken on Earth, only around 20 languages were classified as high-resource. One explanation to their findings can be attributed to the nature of variation in languages found on the Internet. Figure 1.1 is provided to illustrate this point. This abundance of raw digital data in English, Mandarin and Spanish aids and motivates the creation of large annotated NLP datasets in these languages. Therefore, these datasets containing thousands, if not millions, of labeled data-points makes deep supervised methods a viable option in these languages. However, most languages, including many Indian languages, lack the wealth of linguistic data like large monolingual corpora or language specific lexical resources which are necessary for many standard supervised learning techniques used in NLP. These languages termed "low-resource" or "resource-poor" languages require innovative approaches and pose a challenging problem in NLP due to the limited availability of annotated resources. Despite the scarcity of raw digital data, many of these languages are widely spoken around the world. Languages such as Telugu, Punjabi, Bengali, Javanese, Vietnamese, Malayalam and Tamil are used as the sole means of communications by over 2 billion people.
Recently, low-resource NLP research is gaining momentum with various exciting projects such as the Crúbadán project\footnote{http://crubadan.org/}, Leipzig Corpora Collection\footnote{https://corpora.uni-leipzig.de} and the Human Language Project\footnote{http://human-language-project.eu/}, to name a few, that are dedicated to expanding corpora for low-resource languages. Furthermore, conferences such as The International Conference on Language Resources and Evaluation (LREC) are organized with the explicit interest in expanding low-resource corpora for many languages. However, as mentioned before, we cannot depend entirely on the availability of annotated corpora to perform NLP tasks in a low-resource setting.

The lack of annotated data in many languages has been the driving force behind the use of unsupervised learning techniques that does not depend on labeled datasets. These techniques are motivated by the underlying observation that although annotated data is hard to obtain, there are often raw unannotated datasets available which can be exploited. But these techniques were shown to have produced unsatisfactory performance and they lag behind supervised techniques for many tasks. Therefore, low-resource languages are in a dire need of novel techniques to overcome the paucity of available data so that the advancements in NLP can be enjoyed for all languages.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{usage_of_language.png}
\caption{Languages on the Internet (source: Scroll.in)}
\end{figure}
1.1.3 The Problem with Cross-Lingual Resources

Another approach, which is by far the most promising in addressing the data scarcity issue in low-resource languages, is the use of auxiliary cross-lingual resources such as bilingual lexicons, multilingual parallel corpora or dictionaries that can be leveraged to overcome the language barrier. With the growing quantity of raw multilingual text available on the internet with multilingual websites such as Wikipedia\(^4\) and huge digital archives of translated books and news stories, unannotated parallel texts became more widely accessible. The core idea is that this plethora of parallel data can be harnessed to transfer the information from a resource-rich language to a resource-poor language.

More specifically, one variant of this approach uses the multilingual parallel text for training Machine Translation systems. This leads to a naive solution to address the data scarcity problem in low resource languages by employing automatic translation systems like Google Translate\(^5\) for many NLP tasks in a low resource setting. The underlying idea is to translate the text in the low-resource language to a high-resource language which has pre-trained NLP models and tools available to perform a task. But this approach is susceptible to the availability and the accuracy of the translation service, which in most cases is unsatisfactory. To make matters worse, even the best translation services today have difficult in translating idioms and colloquialisms.

Transfer learning, or more specifically, cross-lingual transfer learning strategies provide an alternative approach in solving NLP tasks in a low-resource setting. These approaches are made possible with the undertaking of various crowd-sourced projects such as the PanLex Project\(^6\) or the Wiktionary Project\(^7\) which provides cross-lingual resources like dictionaries, wordnets and other lexical resources for low-resource languages through collaborative efforts of volunteers. The process of cross-lingual transfer learning refers to transfer of information from a resource-rich source language to resource-poor target language with cross-lingual resources like dictionaries acting as the bridge between the languages. However, in most cases, these cross-lingual resources are limited only to lexical components (such as word alignments) without any deeper, more involved high level annotations. Unfortunately, the curation of such cross-lingual resources is both a time and a labour intensive task. Hence, there is a need for architectures that can perform unsupervised transfer learning in the absence of such cross-lingual resources.

---

\(^4\)https://www.wikipedia.org/

\(^5\)https://translate.google.com/

\(^6\)https://panlex.org/

\(^7\)https://en.wiktionary.org/wiki/Wiktionary
1.2 Research Questions

Driven by the three main motivations described in section 1.1, the work presented in this thesis attempts to answer the following research questions:

1. How can we solve NLP tasks in a low-resource scenario where we have limited amounts of labeled data available without the use of any auxiliary cross-lingual resources or any form of cross-lingual supervision?

2. Can we extend the previous question to a scenario where we do not have any labeled data in a low-resource language? The previous scenario assumes that even in low resource languages, there are few labeled data-points available, which is not always the case.

3. Cross-lingual transfer learning strategies in NLP usually involves one source language and one target language. Can we extend this situation by leveraging multiple source languages so that more information can be extracted from different languages that can be transferred to the target low-resource language?

1.3 Main Contributions

To summarize, the main contributions of this thesis are presented below:

1. We present a novel multi-lingual transfer learning approach to solve the data scarcity problem in low resource languages that harness labeled datapoints in multiple resource rich languages to improve the performance of a low-resource language without cross-lingual supervision. In many cases, our experiments show that the improvement in performance is not limited to just resource-poor languages but for many resource-rich languages as well.

2. We provide experimental results of the proposed approach in two different settings characterized by the amount of labeled data available in the low resource language:
   - **Low-resource Setting**: Where labeled data in the target resource-poor language is available in limited amounts (Few-shot Learning Task).
   - **No-resource Setting**: Where there is no labeled data available in the target resource-poor language (Zero-shot Learning Task).

3. We evaluated the robustness of the proposed approach on two well-defined NLP problems:
   - **Sentiment Analysis**: For classification problems like Sentiment Analysis we introduced the Language Invariant Sentiment Analyzer (LISA) architecture that out-performed the previous state-of-the-art approaches on all the languages in the Multilingual Amazon Text Classification dataset. The approach also achieved significant performance gains over prior research on the low-resource Telugu Sentiraama corpus.
• **Event Detection**: For sequence labeling tasks like Open Domain Event Detection, we introduced the MultiLingual Sequence Tagger (M-LiST) architecture that achieved state-of-the-art performance on three low-resource languages in the TempEval2 corpus.

4. Although the scope of this thesis is limited to the two tasks described above, we are confident that the proposed approach can be extended to a variety of NLP tasks for many low-resource languages.

### 1.4 Thesis Organization

The work presented in this thesis is organized as follows:

- **Chapter 1** gives a brief introduction on the importance of labeled corpora for deep learning in NLP. We address the motivation behind our work and defines the problem statement and main contributions of the thesis.

- **Chapter 2** provides a brief overview of RNN-based architectures that we used in our experiments and a primer on transfer learning methodologies and prior research in the domain of low-resource NLP.

- **Chapter 3** describes how representations for different words in different languages are learnt and how to align them in a common multi-lingual embedding space.

- **Chapter 4** introduces the LISA architecture and describes the methodology behind using it for a classification task like Sentiment Analysis.

- **Chapter 5** presents the M-LiST architecture and describes the methodology behind using it for a sequence labeling task like Open Domain Event Detection.

- **Chapter 6** addresses the advantages and shortcomings of the proposed approach and states our concluding remarks and discussions on future work in this direction.
Chapter 2

Background and Related Works

“If I have seen further than others, it is by standing upon the shoulders of giants.”

— Sir Isaac Newton

The contents of this Chapter provides the background necessary to understand the core components of our proposed multilingual transfer learning approach for performing NLP tasks for low resource languages. It also describes the various methodologies of Transfer Learning and, more specifically, Cross-Lingual Transfer Learning strategies used by prior research for NLP in low resource languages.

2.1 Background

In this section, we provide a brief overview of the architectures that we employed for our experiments. Recurrent Neural Networks (RNNs) [39], Long Short Term Memory (LSTMs) [52], Bidirectional Long Short Term Memory (Bi-LSTMs) [49] and Gated Recurrent Units (GRUs) [23] are different types of Neural Networks that take advantage of sequential information that are intrinsically present in languages. We chose these architectures for our experiments because they are able to capture the contextual information in written text that traditional neural networks are incapable of. In a broad sense, these neural architectures can be thought of as a memory mechanism that stores information about the context.

Traditional Neural Networks and Convolutional Neural Networks (CNNs) are also incapable of handling variable length input. They perform well only when the inputs and outputs are of predetermined sizes. The architectures that we experimented with do not suffer from this limitations because they facilitate variable length inputs and outputs. In the following subsection we describe each of them with a bit more detail:
2.1.1 Recurrent Neural Network (RNN)

Figure 2.1 RNN Architecture

Recurrent Neural Network (RNN) is a type of Neural Network dedicated to processing sequential data in which the order of the words in the sentence are considered. A traditional RNN scans the data from left to right where the output from the previous time step is provided as the input to the current time step. RNNs maps a sentence (which is a sequence of words) to a sequence of hidden states which are then used for word level classification. The most important feature of RNNs is its "hidden states" which stores information about the sequence. Figure 2.1 is provided to show the unrolling of the RNN architecture where:

- $x_t$ denotes the input at time step $t$. $X_t$ is usually the one-hot encoding vector corresponding to a particular word in a sentence.
- $a_t$ denotes the hidden state at time step $t$ which can be understood as the memory unit of the RNN network.
- $W_{xh}$ denotes the trainable weights for the connection from the input layer to the hidden layer.
- $W$ denotes the trainable weights for the connection from a hidden layer to another hidden layer.
- $W_{hy}$ denotes the trainable weights for the connection from the hidden layer to the output.
- $y_t$ denotes the output at step step $t$. 

The hidden state \( a_t \) at time \( t \) is calculated as a function of the previous hidden state \( a_{t-1} \) and the input \( x_t \) at the current time step with the following equation:

\[
a_t = f(W_{xh} x_t + W_{a} a_{t-1})
\]

(2.1)

The function \( f \) is usually a nonlinear function such as sigmoid or tanh. The output \( y_t \) at time \( t \) is calculated as a function of the current hidden state \( a_t \) with the following equation:

\[
y_t = W_{hy} a_t
\]

(2.2)

In RNNs, the weights \( W, W_{xh}, W_{hy} \) are shared across all time steps which reflects the fact that the RNN computes the same operations at each time step. This greatly reduces the number of trainable parameters to be learnt by the model as compared to a traditional Neural Network which has different trainable parameters at each layer. As with any Neural Network, the RNN is trained through backpropagation.

### 2.1.2 Long Short Term Memory (LSTM)

Although the RNN architecture was developed to capture the sequential information in sentences, in practice, they fail when the sentences are too long and are not able to capture long-range dependencies between words. This is caused due to the infamous Vanishing and Exploding Gradient problem [13, 94]. Therefore, the LSTM cell was developed to address this problem by extending the basic RNN cell by storing information over long periods of time with the help of a cell state and an efficient gating mechanism.

![Figure 2.2 LSTM Cell](image-url)
Fundamentally, LSTMs have a very similar architecture to RNNs. The only difference is that LSTMs have the ability to add or remove information to a separate cell state with the help of a gating mechanism comprised of an input gate, output gate and a forget gate that controls the flow of information. Figure 2.2 depicts the structure of the LSTM cell. The following equations show how the cell state \( C_t \) and the hidden state \( h_t \) is computed at time step \( t \) :

\[
f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}
\]

\[
i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}
\]

\[
o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.5}
\]

\[
\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.6}
\]

\[
C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.7}
\]

\[
h_t = o_t \odot \tanh(C_t) \tag{2.8}
\]

where,

- \( x_t \) denotes the input the input at time step \( t \).
- \( h_t \) denotes the hidden state at time step \( t \).
- \( C_t \) denotes the cell state at time step \( t \).
- \( f_t \) denotes the output of the forget gate at time step \( t \).
- \( i_t \) denotes the output of the input gate at time step \( t \).
- \( o_t \) denotes the output of the output gate at time step \( t \).
- \( C_t \) denotes the cell state at time step \( t \).
- \( W \) denotes the weight matrix for a particular connection.
- \( b \) denotes the biases associated with a particular connection.
2.1.3 Gated Recurrent Unit (GRU)

The structure of a GRU cell is very similar to that of a traditional LSTM cell but with a different gating mechanism. In a GRU cell, the input gate and the forget gate is merged together to form an update gate. Therefore, instead of deciding what information to forget and what information to add separately, the update gate of the GRU cell makes those decisions together. The GRU cell also merges the cell state and the hidden state. The following equations show how the hidden state at time $t$ is calculated:

\begin{align*}
    z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
    r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
    \tilde{h}_t &= \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \\
    h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{align*}

where,

- $x_t$ denotes the input the input at time step $t$.
- $h_t$ denotes the hidden state at time step $t$.
- $W$ denotes the weight matrix for a particular connection.
- $b$ denotes the biases associated with a particular connection.
- $r_t$ denotes the output of the update gate at time step $t$.
- $z_t$ denotes the output of the output gate at time step $t$. 

Figure 2.3 GRU Cell
2.1.4 Bi-directional Long Short Term Memory (Bi-LSTM)

![Bi-directional LSTM Architecture](image)

Figure 2.4 Bi-directional LSTM Architecture

The one drawback of the previously discussed architectures is that all of them process a sequence from left to right. Therefore, they use information that is present earlier in a sequence to understand the context. Hence, the hidden states and/or cell states at any time step will contain information from earlier in the sentence and not the information that is available later in the sentence.

To address this issue, a more refined version of the LSTM architecture was introduced called the Bi-directional LSTM (Bi-LSTM) [49]. Bi-LSTMs (depicted in Figure 2.4) can be understood as two LSTMs that are stacked together. The most important feature of a Bi-LSTM is its ability to capture the context in a bi-directional fashion. The underlying idea of a Bi-LSTM is that the hidden state and/or cell state at time \( t \) not only depends on the previous inputs in the sequence, but also future inputs. Bi-LSTMs are shown to have superior performance for many NLP tasks such as Question Answering [24] and Dependency Parsing [125].

For each input \( x_t \) at time step \( t \), the Bi-LSTM network employs a forward LSTM and a backward LSTM that computes two hidden states: \( h^f_t \) and \( h^b_t \). The forward LSTM process the sequence from left to right and computes \( h^f_t \) that stores the context to the left of each input \( x_t \). The backward LSTM process the sequence from right to left and computes \( h^b_t \) that stores the context to the right of each input \( x_t \). Then a final hidden state \( h_t \) is computes as the concatenation of the forward and backward hidden states: \( h_t = h^f_t \bullet h^b_t \) where \( \bullet \) represents vector concatenation. The concept of Bi-directional LSTM can be extended to other RNN architectures such as Bi-GRU or Bi-RNN.
2.2 Transfer Learning

Machine Learning algorithms and deep learning architecture have set the benchmark, achieving state-of-the-art performance with impressive accuracy scores and outperforming several baselines for many tasks in various domains. But, more often than not, their achievements are limited to the specific dataset that they were trained on, or the specific task they were learning to solve. Therefore, their performance degrade when they are employed in a different domain or used for a different task (even though the new domain or the new task might be similar to the ones they were trained on). Therefore, these models need to be trained again and built from scratch because they are incapable of retaining information that can be potentially be useful to perform another task in a different domain.

The main motivation behind transfer learning methods is to circumvent this paradigm of isolated learning by transferring the knowledge across different models. The concept of transfer learning was first introduced in the Neural Information Processing Systems (NIPS) workshop\(^1\) titled “Learning to Learn: Knowledge Consolidation and Transfer in Inductive Systems” in 1995. Goodfellow et al. in their book titled "Deep Learning" \([46]\) defined transfer learning as "A situation where what has been learned in one setting is exploited to improve generalization in another setting."

Transfer learning strategies leverage the information learnt by one model and uses it in a different domain or to solve a different task. This is especially useful in a scenario where we have large amounts of labelled data for a task \(T_1\), which we can use to train a model, and generalize the learnt knowledge (in the form of features or weight), which can then be used for another task \(T_2\) that does not have any labelled data available.

2.2.1 Transfer Learning: Formal Definition

Although Goodfellow et al. provided a simple definition of transfer learning that was easy to understand, a more mathematically rigorous definition was provided by Pan et al. in their publication titled "A Survey on Transfer Learning" \([93]\). They established a framework for understanding transfer learning that consists of:

1. A domain \(D\) which is defined as a set consisting of a feature space \(\mathcal{X}\) and a marginal probability distribution \(P(X)\) where,

   - \(D = \{\mathcal{X}, P(X)\}\)
   - \(X = \{x_1, \ldots, x_n\}\)
   - where \(x_i \in \mathcal{X}\) is a sample data point.

---

\(^1\)https://plato.acadiau.ca/courses/comp/dsilver/NIPS95_LTL/transfer.workshop.1995.html
2. A task $T$ which is defined as a set consisting of a class label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ where,

- $Y = \{y_1, \ldots, y_n\}$
- $y_i \in \mathcal{Y}$ where $y_i$ is a class label.

3. Given a source domain $D_S$, a source task $T_S$, a target domain $D_T$ and target task $T_T$ where $D_S \neq D_T$ or $T_S \neq T_T$, transfer learning is defined as a function that learns the conditional probability distribution $P(Y_T|X_T)$ in the target domain $D_T$ by leveraging the information that was learnt from solving $T_S$ on $D_S$.

### 2.2.2 Transfer Learning: Methodologies

![Categories of Transfer Learning Strategies](image)

**Figure 2.5** Categories of Transfer Learning Strategies

Depending upon the amount of labelled data that is available in the source and target domains, the similarities of the domains and the task to be solved, transfer learning techniques can be broadly classified into three major groups (Figure 2.5):
Inductive Transfer Learning: Inductive transfer learning strategies are used in a setting where we have labelled datapoints available for the target task and the source domain is similar to the target domain but the source task is different from the target task (i.e, \( D_S = D_T \) and \( T_S \neq T_T \)). These algorithms leverage the labelled data for the target task and the unlabelled data in the source domain to learn \( P(Y_T|X_T) \) in \( D_T \). For example, [59] developed a heuristic model that identifies misleading training data points in the source domain from the conditional probabilities \( P(Y_T|X_T) \) and \( P(Y_S|X_S) \). [77] used the unlabelled datapoints in the source domain to select which datapoints to be labelled in the target domain using active learning. TrAdaBoost [28] extended the AdaBoost algorithm to utilize datapoints in multiple domains. [45] trained two models in the source and target domains and used a weighted Ensemble learning technique to use them in tandem by assigning weights according to performance of the two models on the datapoints for the target task. [129] used the source domain datapoints to improve the performance of a Support Vector Machine trained in the target domain. Based on the availability of labelled data in the source domain, inductive transfer learning strategies are further divided into two categories:

1. Multi-Task Learning: This form of inductive transfer learning is utilized in a setting where there is an abundance of labeled data in the source domain for the source task which can then be used to transfer knowledge for increasing the performance of the target task. For example, [73] developed a convex optimization algorithm to learn feature weights and meta-priors from related tasks which can be used across different tasks. [57] trained an SVM on related tasks to learn task-invariant features which was then extended by [104] by learning which kernel to use depending on the task. [4] introduced a sparse-feature learning algorithm for training multiple tasks simultaneously. [71] proposed the MT-IVM algorithm that attempts to learn the parameters of a Gaussian Process for multiple tasks where the same Gaussian Prior was assumed for all the tasks. Similarly, [18] also assumed that all the tasks share the same Gaussian Prior but used a covariance matrix to learn the inter-task relationships.

2. Self-Taught Learning: In this setting, we assume that there is no labelled data available in the source domain or the target domain. The work presented in [102] was the first self-taught learning approach which learnt unsupervised features using sparse coding [72] for transferring information across tasks. Recent research in self-taught learning have, however, gravitated toward manifold learning techniques. [124] used Procrustes analysis to transfer information across tasks with the help of aligned manifolds. [86] used concepts from First Order Logic to build Markov Logic Networks for transfer learning. [32] extended the work presented in [86] by using Second Order Markov Logic to transfer knowledge.
Transductive Transfer Learning: Transductive transfer learning strategies are used in a setting where we have labelled datapoints available for the source task and the source task is similar to the target task but the source domain is different from the target domain (i.e, $D_S \neq D_T$ and $T_S = T_T$). Transductive transfer learning for NLP tasks are sometimes referred to as domain adaptation. In domain adaptation tasks, the marginal probabilities of the source domain and target domains are different ($P(X_S) \neq P(X_T)$). For example, a dataset of sentiment labelled movie reviews would have a different marginal distribution than a dataset containing sentiment labelled product reviews. Even though the task is the same (sentiment analysis), the domains are different. Most of the techniques used in domain adaptation tasks leverage the labelled datapoints available in the source domain to improve the performance on the target domain.

[110] proposed the KLIEP algorithm that minimizes the KL-divergence between the source and target marginal distributions. [27] extended the Naive Bayes classifier for transductive transfer learning tasks. [16] used the unlabelled datapoints from the target domain for Structured Correspondence Learning to extract features from the source domain. [30] developed a kernel mapping function that projects the datapoints from the source and target domains to a common high-dimensional vector space and trained a classifier using features from the common vector space. [26] utilized a co-clustering algorithm to propagate the label information from the source domain to the target domain. [135] leveraged labelled and unlabelled datapoints from different domains to build a cross-domain classifier based on Probabilistic Latent Semantic Analysis (PLSA) called ”Topic-Bridged PLSA”. [92] used the Maximum Mean Discrepancy Embedding method, a dimensionality reduction technique, to project the source and target domains to a low dimensional space for transfer learning. Our proposed multilingual transfer learning method is, in part, inspired by the work presented in [44] which employed ”Domain Confusion” to learn domain-invariant features that can be utilized across domains.

Unsupervised Transfer Learning: Unsupervised transfer learning strategies are used in a setting similar to inductive transfer learning where the source domain is similar to the target domain but the source task is different from the target task (i.e, $D_S = D_T$ and $T_S \neq T_T$). However, in an unsupervised transfer learning setting we assume that we have no labelled datapoints in the source domain or the target domain. Prior research in this setting is limited and have resorted to unsupervised approaches such as clustering or dimensionality reduction. Dai et al. proposed a new Self-Taught Clustering (STC) algorithm [29] which tries to learn a shared vector space across domains that is used to cluster small amounts of unlabelled data in the target domain using the unlabelled data in the source domain. Similarly, [127] developed a new dimensionality reduction technique called Transferred Discriminative Analysis (TDA) by first utilizing unsupervised clustering methods to generate, what the authors refer to as, ”pseudo-class labels” for the unlabelled datapoints in the target domain. It then projects the pseudo-class labelled data points in the target domain and the unlabelled datapoints in the source domain to a shared low-dimensional feature space.


2.3 Cross-Lingual Transfer Learning

The process of cross-lingual transfer learning (CLTL) refers to the transfer of resources, labels or trained models from a resource-rich source language to a resource-poor target language. In the framework of transfer learning strategies, cross-lingual transfer learning is a type of transductive transfer learning where the domains are languages. The core idea behind cross-lingual transfer learning strategies is that there are certain concepts that are common in all languages that could be exploited for performing NLP tasks for languages that are low-resource. Cross-lingual transfer learning techniques can be applied in two settings: zero-shot learning or one-shot learning.

One-shot learning (also referred to as few-shot learning) strategies are used in a setting where we have limited amounts of training data available in the low-resource target language. Although, the amount of labelled data available in the low-resource language might not be enough to train a supervised model, it might be adequate to fine-tune a model that was trained on a resource-rich language. One-shot learning tasks presents itself in various real-world scenarios where we do not have enough labelled data for every class or in a setting where new classes are added often. One-shot learning was pioneered by Fei-Fei et al. in their paper titled "One Shot Learning of Object Categories" [42] where they used Bayesian approaches for object classification. Zero-shot learning (also known as Zero-data learning) is a extreme version of one-shot learning where we do not have any labelled data available in the low-resource language. In the following subsections we describe the various techniques used for cross-lingual transfer learning:

2.3.1 CLTL using Machine Translation Systems

The most straightforward approach in CLTL involves using off-the-shelf machine translation systems to translate sentences, words, phrases or documents in the target language to the source language and then learning a classifier in the source language for performing predictions in the target language [62, 122, 123, 10, 78, 19]. The baseline CL-MT [99] method uses this technique by using Google Translate to translate documents in the target language to the source language and learns a classifier in the source language using the bag-of-words features. Similarly, the BiDRL model [138] used Google Translate and employed a joint learning approach to simultaneously learn both word and document representations in both source and target language which are then used for sentiment classification. However, these methods are overly reliant on the performance of the machine translation system utilized, which in many cases, are less than satisfactory.

Another approach utilizing machine translation systems is to train a model on a resource-rich language pair and transfer the weights learnt to a low-resource language pair. For example, [139] trained a “parent” model in a resource-rich language pair, such as French to English, then the trained weights are reused (as weight initialization) for a “child” model trained on a low-resource language pair, such as Turkish to English.

---

2https://translate.google.com/
2.3.2 CLTL using Cross-lingual Word Embeddings

Cross-lingual word embeddings have been successfully applied for cross-lingual transfer learning for many tasks such as cross-lingual dependency parsing [137, 3, 114], cross-lingual document classification [66], cross-lingual semantic parsing [34]. The underlying idea is to project the words in different languages to a shared embedding space. [41] used cross-lingual word embeddings for an active learning approach for cross-lingual named entity recognition by using a shared data-selection policy across languages. [140] improved on the phrase-based decoder of [68] for machine translation by using cross-lingual word embeddings as an added feature for the decoder RNN. Cross-lingual word embeddings are also useful for code-mixed data where a document might contain words from several languages. For example, [34] used cross-lingual word embeddings to build a semantic parser for English-German code-mixed data.

The concept of cross-lingual word embeddings can also be extended to cover multiple languages. Joint Multilingual Training (also referred to as Polyglot Learning) projects words in multiple languages to a shared embedding space which can then be used to train a single model that enables parameter sharing between languages. For example, [2] used multilingual word embeddings to develop a multilingual dependency parser that is capable of parsing multiple languages. The XLM project\(^3\) by Facebook Research is another example of leveraging multilingual word embeddings. XLM makes it possible to train a single model for many languages. It was trained on 2.5 TB of CommonCrawl data and in a hundred languages. The topic of multilingual word embeddings will be discussed further in detail in Chapter 3.

2.3.3 CLTL using Cross-lingual Resources

Most popular methods in CLTL makes use of cross-lingual resources to bridge the language barrier and induce inter-language correspondence. [12] used a bilingual dictionary to translate documents in the target language to the source language and trained a classifier in the source language for text classification. [85] used a bilingual lexicon to translate subjective words and phrases in the source language into the target language. [106] utilizes a bilingual dictionary to translate the classification model from a source language to a target language rather than the documents themselves. [9] used WordNet senses as features for CLTL in Indian languages (Hindi and Marathi). The CLMM model [84] treated the source language and the target language words in an unlabeled bilingual parallel dataset as generated simultaneously by a set of mixture components. The CR-RL approach [130] learned word embeddings by using a set of bilingual word pairs where one part of the word vector contains language specific features and the other part contains language independent features. The CL-SCL model [99] leveraged structural correspondence learning with the help of a bilingual dictionary to learn a source-target feature space. [97] used a parallel corpus between the source language and the target language to learn bilingual paragraph vectors (Bi-PV). The UMM [133] model learned multilingual sentiment-aware word representations based on unlabeled parallel data and used pivot languages to transfer sentiment information in the ab-

\(^3\)https://github.com/facebookresearch/XLM
sence of parallel data. The CLDFA approach [134] adopted cross-lingual distillation and adversarial techniques on parallel corpora for CLTL. Our work draws inspiration from the ADAN-GRL model [22] which employed language adversarial training to learn language invariant features from bilingual word embeddings (BWE) which were created using a parallel corpus. In fact, our proposed approach can be considered as a cross-lingually unsupervised variant of the ADAN-GRL model as we do not rely on parallel corpora to learn word representations. Furthermore, the ADAN-GRL model is limited by the BWE to only incorporate two language pairs (source and target) during training, whereas our proposed approach is capable of leveraging multiple source languages and the target language for adversarial training.

2.3.4 CLTL without Cross-lingual Supervision

Similar to unsupervised transfer learning, unsupervised cross-lingual transfer learning is relatively unexplored. However, recent advancements facilitates CLTL techniques that are not dependent on auxiliary resources such as parallel corpora or bi-lingual lexicons. [63] developed a cross-lingual model without leveraging any form of linguistic knowledge between the source-target language pair, by employing a shared Bi-LSTM to learn language-invariant features using adversarial training that can be utilized across languages. They also trained a private Bi-LSTM to learn language-dependent features, specific to a particular language. In this way, they were able to extract language-features as well as language-specific features. The only drawback of this approach is that it assumes the availability of labelled data in all languages to train the private Bi-LSTM. Similarly, neoteric advances by [21] introduced a shared-private Mixture-of-Experts model (MoE) that learns both language specific features and language invariant features without cross-lingual supervision. Our work presented in this thesis, although related to MoE in objective, with respect to the lack of cross-lingual supervision, differs in the methodology. In Chapter 4 we provide direct comparison of our architecture against MoE (Table 4.4) which indicates that the (language invariant) features extracted by our architecture contains more relevant information than the (language specific + language invariant) features extracted by MoE.
Chapter 3

Multilingual Word Representation

“A word is characterized by the company it keeps.”

— John Rupert Firth

The contents of this chapter provide a primer on the concept of word embeddings and, more specifically, on multilingual word embeddings. This Chapter focuses primarily on the FastText embeddings space and the methodology followed to align multiple FastText embeddings of different languages to a common multilingual embedding space (MUSE embedding space).

3.1 Introduction

One of the first adversities that we faced while developing our proposed multilingual transfer learning approach for low-resource NLP, was the difference in vocabularies of different languages. Different languages have different syntactic and semantic structures which makes it hard to transfer knowledge from one language to another. Therefore, to bridge this language barrier, prior research had resorted to using cross-lingual resources such as bilingual lexicons or parallel corpora for cross-lingual transfer learning.

But, as mentioned in section 1.1.3, building and maintain such cross-lingual resource are time and labour intensive. Therefore, to bridge the language barrier that cross-lingual resources are typically used for, and to alleviate the dependence on cross-lingual supervision for our proposed multilingual transfer learning approach, we aligned the word embeddings of vocabularies of different language into a common multilingual embedding space following the unsupervised variant of the MUSE methodology.
3.1.1 Word Embeddings

Since most of the recent multilingual word embedding techniques used today are based on the concept of monolingual word embeddings, we will first provide a brief review on common methodologies used for building monolingual word embeddings. Most NLP models and algorithms used today require a numerical representation for the words in a sentence as a prerequisite to achieve satisfactory performance. Embeddings algorithms are set of algorithms that learns a function to convert an input set of words or symbol to a out set of vectors. These "embedding" (aka. "vectorizing" or "encoding") methods attempt to map symbolic representations such as words, emojis, character n-grams, dates, etc, to a fixed sized vectors which captures the underlying semantic relation between the words or symbols. Converting words to a vector of number is essential because most machine learning algorithm require a well defined fixed length input.

One of the most primitive methods for vectorizing an input text to a fixed length vector was the Bag-of-Words (BoW) model [50]. It is a popular method and widely used due to its simplicity and the ease with which it could be implemented. The representation of a sentence in the Bag-of-Words model describes it with respect to the occurrence of words within the text. The major disadvantage of the Bag-of-Words model is the fact that it doesn’t capture the information about the structure of the sentence or the order in which the words occur. It is called "bag" of words because it only focuses on whether a set of words occur in the document and not where it occurs in the document.

More recent methods, however, have addressed some of the adversities faced by conventional count based models like the Bag-of-Words model. The pioneering work of the Word2Vec model [88] by Thomas Mikolov in 2013 paved the way for more involved and superior algorithms for learning word embeddings. These methods can be thought of as an extension of the Bag-of-Words model where the underlying idea is based on the Distributional Hypothesis [50]. The Distributional Hypothesis states that the words that are similar, appear in similar contexts. Therefore the meaning of a word can be inferred from the words that surround it. Therefore, each word type can be represented as a vector that captures the semantic information of the words in a local context.

These methods attempt to learn the distributed representation of words by formulating the task as a supervised machine leaning problem where the goal is to predict a given word given context (CBOW), or to predict the context given the word (Skipgram). These models [25, 88, 95] require large monolingual datasets to train on to capture the syntactic and semantic relations between the words in a sentence. Despite the fact that most of the research focused on training word embeddings being relatively new, the use of word embeddings have had widespread applications in many systems to solve various tasks such as Machine translation [80], Natural Language Understanding [25], Dependency Parsing [38] and Sentiment Analysis [108].
Trained word embeddings are evaluated either intrinsically or extrinsically. The intrinsic approach details calculating a similarity measure, such as the cosine similarity metric, to quantify the similarity between two word embeddings. Ideally, words that are similar would have their corresponding word vector close in the embedding space and word that are dissimilar would have embeddings that are far away from each other. In the extrinsic approach, word embeddings are evaluated on its performance on downstream tasks such as Machine Translation.

3.1.2 Multilingual Word Embeddings

Although word embeddings have found widespread applications in many domains, these models are typically restrictive in the sense that it is used to generate representations of words only in the language that they were trained on. The abundance of resource, benchmark datasets and training datasets in the English language lead to a disproportionate amount of focus on building word embeddings in English. This bias towards the English language leads to the negligence of the vast amounts of other less resourceful languages that are spoken around the world. This lead to a situation where novel NLP research in low-resource languages are hindered by the lack of good quality word embeddings in that languages that are imperative for satisfactory performance. Therefore, extending monolingual word embeddings to multilingual word embeddings is gaining momentum and has produced state-of-the-art performance for many tasks in a low-resource setting.

Similar to monolingual word embeddings, multilingual word embeddings represent words in an n-dimensional vector space. The only difference is, multilingual word embeddings are able to represent words from multiple languages into the same embedding space. The underlying idea behind this shared embedding vector space is that there are common “concepts” that are present across different languages. For example, the word “flower” in English has the same meaning as the word “fleu” in French. Therefore, in theory, multilingual embedding spaces tries to embed these concepts that are the same across languages. Although in practice, it is much harder since there are plenty of words in one languages that cannot be directly translated to another language since it does not have any meaningful counterpart in the other language.

Multilingual word embeddings might have little use on its own, but it is essential for many downstream tasks such as Machine Translation. Despite not being a downstream task, learning multilingual word embeddings can be extremely helpful for many NLP applications performed in a cross-lingual transfer learning or a multilingual transfer learning setting. It allows lexicons from different languages to be represented in a common shared vector space that is capable of capturing the semantic and syntactic relations between the word from multiple languages in a cross-lingual fashion.
Multilingual or cross-lingual word embeddings is a well researched problem with a multitude of approaches used to align word embeddings of different languages into a joint embeddings space. Most of these approaches [79, 119, 121, 48] leverage some form of bilingual resources such as parallel text or word alignments to bridge the language barrier posed by the words in different languages. An example is depicted in Figure 3.1 which is presented by Luong et al. in 2015 [79] which align words from English and German in a common embeddings space. Without going into much details, given below are the four main categories that deals with multilingual word embedding methodologies in prior research:

1. **Joint-loss optimization methods** [66, 79, 107]: These methods consists of training models on parallel or monolingual datasets where the goal is to jointly optimise a combination of monolingual and cross-lingual losses over three terms (the source language, the target language, and a regularization term)

2. **Monolingual mapping methods** [87, 11, 3]: These methods initially train monolingual word embeddings for different languages on large monolingual corpora and then learn a linear mapping between them.

3. **Cross-lingual training methods** [51, 97, 67]: These methods train word embeddings for different languages on a parallel corpus and optimize a cross-lingual loss function which encourages the embeddings of similar words in different languages to be close to each other in the shared vector space and encourages the word embeddings of dissimilar word to be far away from each other.
4. **Pseudo cross-lingual training methods** [36, 131, 120]: These methods create a code-mixed corpus by mixing contexts of different languages in the same dataset and then training an off-the-shelf word embedding model on this newly created corpus. The underlying idea is that this method could capture cross-lingual relations from the cross-lingual contexts.

For the purpose of our proposed approach, our goal is to learn a common embedding space consisting of the resource-rich source languages and the resource-poor target language. Equipped with such a joint embedding space, we hypothesize that by projecting labelled examples from the resource-rich languages to the joint multilingual embedding space and training a model on them, we simultaneously obtain the ability to perform predictions in all the languages including the resource-poor language. In the following Sections we describe the FastText embedding space that we used in all our experiments and the MUSE approach to align the FastText embeddings in different languages to a common shared embedding space.

### 3.2 FastText Embeddings

Over the last decade, with the introduction of Word2Vec [88] and GloVe [95] embeddings, a lot of research was focused on neoteric approaches to developing new embeddings. However, most of these approaches build on Word2vec or GloVe used supervision in the form of external syntactic or semantic knowledge. On the other hand, more recently, there have been developments that are purely unsupervised, the most notable being ELMo [96] and FastText [17].

Similar to Word2vec, FastText involves training Continuous-Bag-Of-Words or Skip-Gram models with or without negative sampling, softmax or hierarchical softmax. FastText word vectors are fast to train and are available in 157 languages trained on texts sourced from Crawl and Wikipedia. The one glaring drawback of Word2vec and Glove was that they were not capable of handling words that were not included in the training corpus. This is because they treated words as an indivisible token while attempting to learn their embedding vector. Therefore, when it encounters an Out-Of-Vocabulary (OOV) word they fail to retrieve their respective embedding vector.

FastText, on the other hand does not suffer from these drawbacks because the FastText methodology treats character n-grams as the smallest indivisible token and not the entire word. Therefore, the underlying idea behind FastText is that instead of learning the embedding for a word directly (as is done in Word2Vec and GloVe), it learns the representation for each character n-gram instead. Each word can then be treated as a bag of character n-grams and the overall vector representation for the word is the sum or average of the vector representation of its character n-grams.

---

For example, let us take the case of the word "flower" to understand how FastText produces its corresponding embedding. We first choose a value for $n$ (say 3) to extract tri-grams from the word. The tri-grams for the word "flower" are 'flo', 'flo', 'low', 'owe', 'wer' and 'er'. And the vector representation for "flower" is the sum of all the vector representation of its tri-grams. The '(', and ')' are special symbols used to represent the beginning of a word and the end of a word respectively. In this way, FastText can generate embeddings for unseen words by adding the vector representations of all its constituent n-grams.

The one major drawback of FastText is that it requires a lot of memory. This is because the model generates embeddings for its characters and not for words. But the authors argue that the memory requirements can be reduced by limiting the n-grams within a minimum and maximum n-gram range and by hashing the n-grams to values within the range 1 to $K$ using a mapping function. The mapping function of choice was the Fowler-Noll-Vo Hashing Function. The authors of FastText proposed two methods (one method applied FastText to learn unsupervised word embeddings and the other method applied FastText to learn a supervised classification task, both of which are available on Github\(^2\)) for training word embeddings in two different publications:

1. **Enriching Word Vectors with Subword Information** \([17]\) : In this paper the authors used the same Skip-Gram model along with negative sampling as that of the original Word2Vec model by Mikolov et al. to score the similarity between the target word $w_t$ and a word belonging to the context $w_c$. The only major difference between this approach and Word2vec was that the similarity function used in Word2vec was a direct dot-product between the word vectors of $w_t$ and $w_c$ and the similarity function used in this paper was the dot product between the two words represented as sums of n-gram vectors.

2. **Bag of Tricks for Efficient Text Classification** \([61]\) : In this paper the authors trained word embeddings as the byproduct of a text classification task where each word $w_t$ was represented as a sum of n-grams and then an embedding was created for the entire document as the average of the word embeddings. This document embeddings was used by a multinomial logistic regression model to predict the label. The overall loss function is given below:

$$
F = -\frac{1}{N} \sum_{n=1}^{N} y_n \log(f(BAx_n))
$$

when $x_n$ represents an n-gram feature, $A$ represents the averaged text embedding and $B$ converts the averaged embeddings to the pre-softmax values for each class label. Given the large number of classes, a hierarchical softmax was applied to reduce the computational complexity.

\(^2\)https://github.com/facebookresearch/fastText
3.3 MUSE: Multilingual Unsupervised and Supervised Embeddings

In a nutshell, the MUSE (Multilingual Unsupervised and Supervised Embeddings) methods can be understood as aligning FastText embeddings of words from different languages to a common joint embedding space. Previous cross-lingual embedding training techniques described in section 3.1.2 rely on cross-lingual resources such as parallel corpora or bilingual lexicons. In contrast, the MUSE approaches do not require any cross-lingual supervision and uses adversarial training techniques to align word embeddings from different languages. MUSE provides two approaches to align word embeddings of two languages:

1. **Supervised Approach**: This approach leverages a bilingual dictionary or an identical character string to learn a linear transformation from the source language to the target language using the iterative Procrustes alignment.

2. **Unsupervised Approach**: This approach does not rely on bilingual dictionaries or an identical character strings. It uses adversarial training to learn a mapping from the source language to the target language using the iterative Procrustes alignment. We use this variant of the MUSE approach in all our experiments.

For our experiments described in Chapter 4 and Chapter 5, we train fastText embeddings to project each word to a monolingual semantic space for each language in all our datasets. We then employ the unsupervised MUSE approach to align the monolingual FastText vectors of each language in an adversarial manner to a common multilingual semantic space. While training MUSE we use English as the target vector space and align all the other monolingual vector spaces to this space. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_a\}$ and $\mathcal{Y} = \{y_1, y_2, \ldots, y_b\}$ be the source and target fastText word embeddings respectively.
Let $W$ be a linear mapping from $\mathcal{X}$ to $\mathcal{Y}$. A discriminator is trained to discriminate between elements randomly sampled from $W\mathcal{X}$ and $\mathcal{Y}$ while $W$ (which acts as the generator) is jointly trained to fool the discriminator. The discriminator loss function $\mathcal{L}_D(\theta_D|W)$ is formulated as:

$$
\mathcal{L}_D(\theta_D|W) = -\frac{1}{a} \sum_{i=1}^{a} \log P_{\theta_D}(\text{source} = 1|Wx_i) - \frac{1}{b} \sum_{i=1}^{b} \log P_{\theta_D}(\text{source} = 0|y_i)
$$

The Mapping objective function used to train $W$ is given by:

$$
\mathcal{L}_W(W|\theta_D) = -\frac{1}{a} \sum_{i=1}^{a} \log P_{\theta_D}(\text{source} = 0|Wx_i) - \frac{1}{b} \sum_{i=1}^{b} \log P_{\theta_D}(\text{source} = 1|y_i)
$$

Where $\theta_D$ denotes the discriminator parameters and $P_{\theta_D}(\text{source} = 1|z)$ is the probability that a vector $z$ is the mapping of a source embedding according to the discriminator.

Next, a synthetic parallel vocabulary consisting of the most frequent words and their mutual nearest neighbors is extracted from the resulting shared embedding space $W$ to fine-tune the mapping using the closed-form Procrustes solution [105] given by:

$$
W^* = \arg\min_{W \in O_d(\mathbb{R})} \|WX - Y\|_F = UV^T \\
\text{with } U\Sigma V^T = \text{SVD}(YX^T)
$$

Where $X$ and $Y$ are two aligned matrices containing the embeddings of the words in the trained space $W$, $d$ represents the dimension of the embeddings, $O_d(\mathbb{R})$ is the space of $d \times d$ matrices of real numbers with the orthogonality constraint and $\text{SVD}(YX^T)$ represents the singular value decomposition of $YX^T$. 
3.4 Conclusions

In this Chapter, we provided a basic introduction to the notion of word embeddings and multilingual word embeddings. We then proceeded to describe the FastText embedding space and how FastText embeddings of different languages can be aligned to a common multilingual embedding space using the unsupervised variant of the MUSE approach. The MUSE aligned multilingual word embedding, which are created without the use of any cross-lingual resources, provides a solid foundation to perform our proposed multilingual transfer learning approach for low-resource Sentiment Analysis and Open Domain Event detection which are described in the following Chapters.
Chapter 4

Learning Language Invariant Features for Sentiment Analysis

"With public sentiment, nothing can fail. Without it, nothing can succeed."

— Abraham Lincoln

The contents of this chapter provide a primer on the task of sentiment analysis and describes in detail the various components of the Language Invariant Sentiment Analyzer (LISA) architecture and different experimental set-ups to perform cross-lingual sentiment analysis for low-resource languages.

4.1 Introduction

Sentiment analysis (also known as opinion mining) is a widely researched NLP problem with state-of-the-art solutions capable of attaining human-like accuracies for various languages. Sentiment analysis refers to a series of methods, techniques, and tools aimed at extracting the intended sentiment from a written opinion. Traditional sentiment analysis systems deal with the interpretation and classification of opinions or emotion into three main categories: Positive, Neutral or Negative. Today, sentiment analysis models and architectures are able to detect the sentiment polarity in different forms of written texts such as documents, paragraphs, sentences, or phrases.

With the advent of Web 2.0 which is characterized by the growth of user-generated content and the rise of social media, the identification and extraction of subjective information from this abundance of unstructured data on the internet is vital for business entities to understand the social sentiment of their brand, customers, products or services. Understanding customer’s sentiments are essential for businesses due to the fact that customers are able to express their thoughts and opinions more freely than ever before.
Sentiment analysis is considered a challenging problem in NLP. First of all, finding the sentiment of written text is highly subjective. It is influenced by personal experiences, ideologies, thoughts and beliefs. Therefore, what one person considers a positive statement, might be negative for another. Sentiments and opinions also depend on the context of its occurrence in text. Machine learning techniques do not easily understand the context unless it is provided explicitly. However, analyzing sentiment without context is naive and even dangerous in some cases. Irony and sarcasm pose another challenge to sentiment analysis systems. When it comes to irony and sarcasm, people express their negative sentiments using positive words, which is difficult for computers to detect without having a thorough understanding of the context in which the sentiment was expressed. Furthermore, sentiment analysis systems have trouble dealing with comparative expression like: "This cake is better than the chocolate cake", "This cake is better than nothing" or "This cake is second to none". Without context, it is not easily for computers to understand the sentiment polarity of such sentences.

4.1.1 Sentiment Analysis: Benefits and Use Cases

Off-the-shelf sentiment analysis tools and APIs such as Clarabridge\(^1\), IBM Watson NLU\(^2\), Microsoft Text Analytics API\(^3\) and InMoment\(^4\) simplify and facilitates the use of sentiment analysis in many industrial domains, such as:

1. **Brand monitoring**: By analyzing the key aspects of a brand’s products or services that customers are concerned with and understanding the underlying sentiment towards those aspects from social media, business entities gain an overall idea of the public’s perception of their brand.

2. **Customer service**: By automatically processing user feedback, from survey form responses to social media conversations, businesses are able to cater more efficiently to their customer’s needs, and tailor products and services accordingly.

3. **Social media monitoring**: The 2016 US elections provides a good example of how social media monitoring is influential in shaping the political landscape of an entire country. By monitoring websites such as Facebook and Twitter, political parties are able to understand the voter’s opinion about their candidates and their ideologies which helps them target their voters more effectively.

4. **Market research**: Sentiment analysis can also be used to understand the public opinion of competitors in a particular market. This helps businesses to understand the strengths and weaknesses of their competitors which in turn helps them improve their business and marketing strategies.

\(^1\)https://www.clarabridge.com/
\(^2\)https://www.ibm.com/cloud/watson-natural-language-understanding
\(^3\)https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/
\(^4\)https://inmoment.com/
4.1.2 Different Forms of Sentiment Analysis

Sentiment analysis tools and models assume various forms depending on the task description. From models that extracts sentiment polarity ("positive", "negative" or "neutral") to those that detect emotions ("happy", "sad", "angry", etc), or even identifying intentions ("interested" or "not interested"). Presented below is a non-exhaustive list of the most popular types of sentiment analysis methods used today:

1. **Coarse-grained Sentiment Analysis**: This type of sentiment analysis deals with processing whole documents and sentences to classify it into one of three categories:
   
   (a) Negative
   (b) Neutral
   (c) Positive

   The experiments described in this chapter are conducted to solve a restricted variant of Coarse-grained Sentiment Analysis where we classify documents into two categories (negative and positive).

2. **Fine-grained Sentiment Analysis**: In some cases, the traditional three classes used to classify the sentiment might be too restrictive. Therefore, to gain deeper insights, Fine-grained Sentiment Analysis provides five classes to account for the variation and intensity of the sentiment polarity:
   
   (a) Very negative
   (b) Negative
   (c) Neutral
   (d) Positive
   (e) Very positive

3. **Emotion Detection**: Emotion detection attempts to detect emotions (like happiness, sadness, anger, etc) in written text. Emotion detection is a complex task as different people can express emotions in different ways. Therefore, certain words such as "kill" can be used to represent different emotions depending on context. For example, "You killed him" is an angry statement vs "He is killing it at work!" which is a happy statement.

4. **Aspect-based Sentiment Analysis**: Aspect-based sentiment analysis systems deal with identifying certain aspects in text and then assigning the aspect a sentiment polarity label. For example, in product reviews such as "The built-in camera in this phone is abysmal!", an aspect-based sentiment analysis model would be able to understand the negative opinion about the aspect "camera".
4.1.3 Sentiment Analysis : Low-resource Setting

Traditional sentiment analysis techniques have relied on using supervised term weighting methods including terms’ distribution of classes, word-level polarity scoring and using SVMs [37] and Naive Bayes classifiers [98] for pattern extraction using hand-crafted features. However, with the advancement of deep learning techniques such as LSTMs and CNNs, the ability of algorithms to analyze sentiment in text has improved considerably and has now enabled the extraction of high quality sentiment data from written texts.

One majorly overlooked factor in the performance of these approaches is their dependency on large annotated datasets compiled from multiple data sources related to or sourced from newspapers, tweets, photos and product reviews [108, 64, 112, 55, 126]. The abundance of raw digital data in resource-rich languages aids and motivates the creation of annotated resources for sentiment analysis in these languages. Conversely, the paucity of annotated data in most languages makes it a challenging task to develop deep learning based sentiment analysis tools for them. The lack of annotated datasets have facilitated the use of unsupervised learning techniques for low-resource sentiment analysis such as K-means clustering or lexicon-based methods. Undoubtedly, the performance of unsupervised models are unsatisfactory compared to their supervised counterparts [54]. Hence there is a pressing need to pay special attention to developing solutions capable of sentiment analysis in a low-resource setting.

Cross-lingual sentiment classification (CLSC) methods try to alleviate this problem by leveraging labeled data from one language to improve the performance on another language [12]. But vanilla transfer learning techniques such as training a neural model on one language and applying the trained model on another language via weight sharing, cannot be easily applied for cross-lingual tasks due to the limited overlap between the vocabularies of the different languages and difference in their syntactic structure [22]. However, these methods typically rely on auxiliary cross-lingual resources such as a parallel corpora [136, 134], bilingual lexicons [85] or the use of machine translation systems [62, 123, 99, 19] to bridge the language barrier. Unfortunately, these resources are not readily available for most languages and the curation of such cross-lingual resources is both a time and a labour intensive task. Hence, there is a need for architectures that can perform well in the absence of such cross-lingual resources.

4.1.4 Proposed Approach : Overview

Inspired by the recent work in the domain of cross-lingual transfer learning, in this chapter, we address the problem of low-resource sentiment analysis in the absence of cross-lingual resources by presenting a neural Language Invariant Sentiment Analyzer (LISA) architecture. The proposed architecture is capable of training on multiple monolingual sentiment labelled datasets to learn language agnostic sentiment features that can be transferred to perform sentiment analysis in low-resource languages without leveraging any form of cross-lingual supervision.
We formulate the task as a multi-lingual transfer learning (MLTL) language adaptation problem [136] where we attempt to learn language-agnostic sentiment features via adversarial training on labelled documents \( \{s_1, s_2...s_n\} \) from multiple resource-rich (source) languages to improve the performance on documents \( \{t_1, t_2...t_m\} \) from a low-resource (target) language. The key components of our approach include learning monolingual word embeddings from \( \{s_1, s_2...s_n\} \cup \{t_1, t_2...t_m\} \) and projecting them to a shared multilingual semantic space. We employ an LSTM network to learn latent features \( (z) \) from this multilingual space which is then used by a sentiment classifier \((S_c)\) to predict the sentiment polarity of a document \( d \in \{s_1, s_2...s_n\} \cup \{t_1, t_2...t_m\} \). Concurrently, a language classifier \((C_L)\) is trained to predict the language of document \( d \) based on \( z \). During the adversarial training we try to minimize the binary cross-entropy loss of \( C_S \), while at the same time we maximize the cross-entropy loss of \( C_L \). This results in a setting where the LSTM learns to produce latent features \( (z) \) that predicts the sentiment of document \( d \) correctly independent of the language of document \( d \). We hypothesize that in this setting, the latent features \( (z) \) trained would contain sentiment features that are language agnostic. It should be noted that the terms ”language-invariant” and ”language-agnostic” are used interchangeably in this thesis.

4.2 Dataset Description

We conduct our experiments on two publicly available sentiment classification datasets:

1. The Multilingual Amazon Review Text Classification dataset [99] consists of sentiment labelled data in multiple languages. The vast amount of prior work on this dataset helps us to directly compare our results with the pre-existing state-of-the-art CLSC methods.

2. The Sentiraama Corpus [43] is a real-world low-resource sentiment corpus in Telugu (an agglutinating Indian language). We use this dataset to test the robustness of our system and evaluate our results in a truly low-resource setting.

In the following subsections we describe both the corpora in detail.

4.2.1 Multilingual Amazon Review Text Classification dataset

The Multilingual Amazon Review Dataset contains sentiment labeled product reviews in four languages (English, German, French and Japanese) across three domains (Books, Dvd and Music). The German, French and Japanese reviews were crawled from Amazon and the corpus was enhanced with English reviews from [16]. Each review contains a domain label, a review summary, a review text, and a rating from the set \( \{1, 2, 4, 5\} \) where \( \{1, 2\} \) denotes negative sentiment and \( \{4, 5\} \) denotes positive sentiment. The review summaries provide little information as compared to the review text and hence the review summaries are discarded and not used in our experiments.
The review texts in each domain for each language are split into three disjoint balanced sets, namely, Train set, Test set and Unlabeled set. The dataset statistics are presented in Table 4.1.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Test</th>
<th>Unlabelled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>English</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Books</td>
<td>2000</td>
<td>2000</td>
<td>50000</td>
</tr>
<tr>
<td>DVD</td>
<td>2000</td>
<td>2000</td>
<td>30000</td>
</tr>
<tr>
<td>Music</td>
<td>2000</td>
<td>2000</td>
<td>25220</td>
</tr>
<tr>
<td><strong>German</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Books</td>
<td>2000</td>
<td>2000</td>
<td>165470</td>
</tr>
<tr>
<td>DVD</td>
<td>2000</td>
<td>2000</td>
<td>91516</td>
</tr>
<tr>
<td>Music</td>
<td>2000</td>
<td>2000</td>
<td>60392</td>
</tr>
<tr>
<td><strong>French</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Books</td>
<td>2000</td>
<td>2000</td>
<td>32870</td>
</tr>
<tr>
<td>DVD</td>
<td>2000</td>
<td>2000</td>
<td>9358</td>
</tr>
<tr>
<td>Music</td>
<td>2000</td>
<td>2000</td>
<td>15940</td>
</tr>
<tr>
<td><strong>Japanese</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Books</td>
<td>2000</td>
<td>2000</td>
<td>169780</td>
</tr>
<tr>
<td>DVD</td>
<td>2000</td>
<td>2000</td>
<td>68326</td>
</tr>
<tr>
<td>Music</td>
<td>2000</td>
<td>2000</td>
<td>55892</td>
</tr>
</tbody>
</table>

Table 4.1 Multilingual Amazon Review Text Classification dataset statistics.

4.2.2 Sentiraama Dataset

The Sentiraama dataset consists of sentiment labelled documents in the Telugu language which are organized into four domains: Books, Movies, Products and Song Lyrics. Each document is given a positive or a negative label. The data was crawled from online sources\(^{5-8}\) and ratings above 2.5 were labelled positive and ratings below 2.5 were labelled negative. The corpus statistics are presented in Table 4.2.

\(^{5}\)telugulyrics.org
\(^{6}\)elugu.samayam.com
\(^{7}\)tupaki.com
\(^{8}\)a2zsonglyrics24.blogspot.com
Table 4.2 Sentiraama corpus statistics.

<table>
<thead>
<tr>
<th></th>
<th>Books</th>
<th>Movies</th>
<th>Products</th>
<th>Lyrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>100</td>
<td>136</td>
<td>100</td>
<td>230</td>
</tr>
<tr>
<td>Negative</td>
<td>100</td>
<td>131</td>
<td>100</td>
<td>109</td>
</tr>
<tr>
<td>Total</td>
<td>200</td>
<td>267</td>
<td>200</td>
<td>339</td>
</tr>
</tbody>
</table>

To avoid cross-domain discrepancies we restrict our experiments to the Books and Movies domain as it has similar counterparts in the Multilingual Amazon Review Dataset, i.e, Books and Dvd respectively. We divide the Books and Movie domains of the Sentiraama dataset to create a Train set and a Test set using an 80-20 train-test split. The statistics of the subset of the corpus that are used in our experiments are listed in Table 4.3.

Table 4.3 Subset of the Sentiraama corpus used in our experiments.

<table>
<thead>
<tr>
<th></th>
<th>Books</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+ve</td>
<td>-ve</td>
</tr>
<tr>
<td>Train</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Test</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

4.3 Multilingual Word Representation

For our experiments, the first step was to align the words in different languages to a common multilingual word embedding space. The use of multilingual word representations for words in multiple languages alleviates the dependence on cross-lingual resources in cross-lingual sentiment classification. To this end, we employed the MUSE approach described in Chapter 3 to learn multilingual word embeddings for all the words in the Multilingual Amazon Review Text Classification dataset and the Sentiraama dataset. We note that the pre-trained MUSE embeddings for English, French and German are hosted on GitHub[^9], but the pre-trained models for Japanese and Telugu are not included. Therefore, we trained our own MUSE embeddings by first training FastText representations for all the words in each language and then aligning them to the English embedding space following the unsupervised variant of the MUSE approach. To train the FastText embeddings we used the Train sets for all the languages in both the corpora along with the Unlabelled set for the languages in the Multilingual Amazon Review Text Classification dataset.

[^9]: https://github.com/facebookresearch/MUSE
4.4 LISA Architecture

![Diagram](image)

**Figure 4.1** The Language Invariant Sentiment Analyzer (LISA) Architecture

The input to the LISA model is a review $r_i$ that is made up of a sequence of words $w_1, w_2, \ldots w_k$. Each review $r_i$ has associated with it a language label $l_i \in L$ where $L = \{l_1, l_2, \ldots l_p\}$ is the set of all language labels used in training. Additionally, each review $r_i$ has associated with it a sentiment label $t_i \in \{\text{positive}, \text{negative}\}$ which denotes the sentiment polarity of the review. We project each word $w_i$ to the multilingual semantic space (from section 4.3) to obtain a sequence of $n$-dimensional word embeddings $e_1, e_2, \ldots e_k$ where $e_i \in \mathbb{R}^n$. The following subsections describes in detail the individual components of the LISA architecture. Figure 4.1 shows the overall architecture of the proposed model.

4.4.1 Multilingual Sequence Encoder ($\mathcal{H}$)

The purpose of the Multilingual Sequence Encoder ($\mathcal{H}$) is to processes the sequence of word embeddings $(e_1, e_2, \ldots e_k)$ and transform it into an $m$-dimensional (hidden) vector $\mathcal{H}(r_i)$. To this end, the embeddings for all the words in review $r_i$ are passed sequentially through a Long Short-Term Memory (LSTM) network [52]. LSTMs are a variant of RNNs that leverages sequential information which are inherently encoded in a sentence to learn features that model the long-term dependencies between the words. The LSTM network, at each time step outputs a hidden state $h_i$ for every input word embedding.
\( e_i, \text{ such that: } \)
\[
h_i = \text{LSTM}(e_i, h_{i-1}) \in \mathbb{R}^m
\]

The final hidden state \( \mathcal{H}(r_i) = h_k \) is then passed through a Language Discriminator (\( \mathcal{C}_L \)) and a Sentiment Analyzer (\( \mathcal{C}_S \)).

### 4.4.2 Language Discriminator (\( \mathcal{C}_L \))

![Figure 4.2 LISA : Language Discriminator](image)

The goal of the Language Discriminator (\( \mathcal{C}_L \)) is to predict the language label \( l_i \) based on \( \mathcal{H}(r_i) \). In other words, \( \mathcal{C}_L \) tries to predict the language from which the sequence of words \( w_1, w_2, \ldots w_k \) come from. The \( \mathcal{C}_L \) comprises of a Gradient Reversal Layer (\( GRL_\lambda \)), followed by two Dense Layers which uses the LeakyReLU activation function and an output Softmax Layer that applies the softmax function over all the languages used in training. During backpropagation, \( GRL_\lambda \) multiplies the gradients by a factor of \(-\lambda\) and during the forward pass it acts as the identity function. \( \lambda \) is hyperparameter in the network.

### 4.4.3 Sentiment Analyzer (\( \mathcal{C}_S \))

![Figure 4.3 LISA : Sentiment Analyzer](image)
The Sentiment Analyzer ($C_S$), as the name suggests, tries to predict the sentiment label $t_i$ of the input review $r_i$ based on $\mathcal{H}(r_i)$. The $C_S$ is made up of two Dense Layers which uses the LeakyReLU activation function followed by an output Softmax Layer that applies the softmax function over the two sentiment polarities (positive and negative).

### 4.5 Training Set-Up

Inspired by recent works [47, 44, 14], we train the LISA model using adversarial training on a set of labeled reviews $R = \{r_1, r_2, \ldots, r_n\}$ which belong to multiple languages. The aim of the LISA model is to predict the sentiment label ($t_i$) for a given review ($r_i$) independent of the language label ($l_i$).

We formulate the learning objective in a way that minimizes the sentiment classification loss from $C_S$ and maximizes the language classification loss from $C_L$. As a result, the LISA model tries to jointly optimize the below functions:

1. Minimize the sentiment classification loss:
   \[ \arg\min_{\mathcal{H}, C_S} f_1(C_S(\mathcal{H}(r_i)), t_i) \]  
   \[ (4.1) \]

2. Maximize the language classification loss:
   \[ \arg\max_{C_L} f_2(C_L(\mathcal{H}(r_i)), l_i) \]  
   \[ (4.2) \]

Where $f_1$ denotes the binary cross entropy loss function corresponding to the two labels (positive, negative) produced by sentiment classifier $C_S$ and is formulated as:

\[ f_1(x, y) = -y \log(x) - (1 - y) \log(1 - x) \]  
\[ (4.3) \]

And $f_2$ denotes the cross entropy loss function corresponding to the $L$ labels (English, French, Telugu, etc) produced by the language classifier $C_L$ and is formulated as:

\[ f_2(x, y) = -\sum_{i=1}^{L} y_i \log(x_i) \]  
\[ (4.4) \]

This results in a setting where the $C_L$ tries to predict $l_i$ based on a given $\mathcal{H}(r_i)$ and the encoder $\mathcal{H}$ tries to 'fool' the $C_L$ by learning to create $\mathcal{H}(r_i)$ that is minimally influenced by the language label $l_i$ while at the same time, is maximally influenced by the $C_S$ to predict the sentiment label $t_i$ correctly. With the incorporation of a Gradient Reversal Layer $GRL_\lambda$ [44] between $\mathcal{H}$ and $C_L$, the network learns to correctly predict the sentiment label $t_i$ and at the same time, learns to incorrectly predict the language labels $l_i$. By using $GRL_\lambda$, the optimization functions (equations 4.1 and 4.2) can be simplified as:

\[ \arg\min_{\mathcal{H}, C_S, C_L} f_1(C_S(\mathcal{H}(r_i)), t_i) + f_2(C_L(GRL_\lambda(\mathcal{H}(r_i))), l_i) \]  
\[ (4.5) \]
4.6 Experiments and Results

In this section we present an extensive set of experiments conducted on the Multilingual Amazon Review Text Classification dataset and the Telugu Sentiráama sentiment classification corpus. We evaluate our approach in the two settings described below:

1. **Low-resource setting**: We evaluate the performance of the LISA architecture in the low-resource setting (termed LISA-LR) by training it on the Train sets from multiple source languages and the limited Train set in the target language and then testing on the Test set of the target language. We perform experiments in this setting to evaluate the performance of the LISA model for few-shot learning tasks.

2. **No-resource setting**: In the no-resource setting, we assume that the training data is not available for the target language. We train the LISA model (termed LISA-NR) on the Train sets of the source languages and evaluate the model on the target language Test set. We perform experiments in this setting to evaluate the performance of the LISA model for zero-shot learning tasks.

To show the effectiveness of the Language Discriminator ($C_L$), we conducted ablation experiments in the low-resource setting where we remove $C_L$ from the LISA architecture. In this variant of the LISA model (termed LISA-NoLD), the Sentiment Analyzer ($C_S$) only depends on the MUSE embeddings to learn $H(r_i)$ to learn sentiment features. We perform the ablation experiments with LISA-NoLD to show that the Language Discriminator ($C_L$) is a significant component of the LISA architecture and that the use of MUSE embeddings is not the sole cause of performance gains presented in Table 4.4 and Table 4.5. Our experiments show that LISA-LR performs significantly better in most cases than LISA-NoLD.

4.6.1 Results for the Amazon Dataset

For the Multilingual Amazon Review Text Classification dataset in the low-resource setting, we train LISA-LR on the Train sets of all the four languages. We then test it on the Test set of the target language. In the no-resource setting, we train LISA-NR on the Train sets of three languages and test it on the Test set of the fourth language. We do this for each domain in the corpus independently.

We compare our results against prior state-of-the-art methods on this dataset presented below:

1. Methods that use Machine Translation systems:
   
   • **CL-MT**: The baseline CL-MT [99] method uses Google Translate\[^{10}\] to translate documents in the target language to the source language and learns a classifier in the source language using the bag-of-words features.

[^10]: https://translate.google.com/
• **BiDRL**: Similarly, the BiDRL model [138] used Google Translate and employed a joint learning approach to simultaneously learn both word and document representations in both source and target language which are then used for sentiment classification.

2. Methods that rely on auxiliary cross-lingual resources:

• **UMM**: This method [133] attempts to learn multilingual sentiment-aware word representations based on unlabeled parallel data. In the absence of parallel data, the UMM uses pivot languages to transfer sentiment information from one language to another.

• **Bi-PV**: This method [97] uses a parallel corpus between the source language and the target language to learn bilingual paragraph vectors (Bi-PV) for sentiment classification.

• **CR-RL**: The CR-RL approach [130] learned word embeddings by using a set of bilingual word pairs where one part of the word vector contains language specific features and the other part contains language independent features.

• **CL-SCL**: The CL-SCL model [99] leveraged structural correspondence learning with the help of a bilingual dictionary to learn a source-target feature space for sentiment analysis.

3. **MAN-MoE**: This method [21] does not rely on cross-lingual supervision. This approach alleviates the need for cross-lingual resources by introducing a shared-private Mixture-of-Experts model (MoE) that learns both language specific features and language invariant features without cross-lingual supervision.

The results are presented in Table 4.4.

<table>
<thead>
<tr>
<th></th>
<th>German</th>
<th>French</th>
<th>Japanese</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Books</td>
<td>DVD</td>
<td>Music</td>
</tr>
<tr>
<td>CL-MT</td>
<td>79.68</td>
<td>77.92</td>
<td>77.22</td>
</tr>
<tr>
<td>BiDRL</td>
<td>84.14</td>
<td>84.05</td>
<td>84.67</td>
</tr>
<tr>
<td>UMM</td>
<td>81.65</td>
<td>81.27</td>
<td>81.32</td>
</tr>
<tr>
<td>Bi-PV</td>
<td>79.51</td>
<td>78.60</td>
<td>82.45</td>
</tr>
<tr>
<td>CR-RL</td>
<td>79.89</td>
<td>77.14</td>
<td>77.27</td>
</tr>
<tr>
<td>CL-SCL</td>
<td>79.50</td>
<td>76.92</td>
<td>77.79</td>
</tr>
<tr>
<td>MAN-MoE</td>
<td>82.40</td>
<td>78.80</td>
<td>77.15</td>
</tr>
<tr>
<td>LISA-LR</td>
<td><strong>85.45</strong></td>
<td><strong>84.90</strong></td>
<td><strong>86.55</strong></td>
</tr>
<tr>
<td>LISA-NR</td>
<td>55.60</td>
<td>55.50</td>
<td>58.90</td>
</tr>
<tr>
<td>LISA-NoLD</td>
<td>81.20</td>
<td>77.70</td>
<td>80.75</td>
</tr>
</tbody>
</table>

*Table 4.4* Results on the Multilingual Amazon Review Text Classification dataset. The numbers denote binary classification accuracies.
4.6.2 Results for the Sentiraama Corpus

The underlying idea behind the experiments on the Sentiraama corpus is to leverage the sentiment labelled data in multiple languages from the Amazon dataset and transfer that information to perform sentiment analysis on the Sentiraama data. For both the settings presented below, we ran experiments for the Books and Movies domain separately. We evaluate the results of LISA-LR, LISA-NR and LISA-NoLD against the Bernoulli Naive Bayes [103] and SVM [60] baselines that use TF-IDF features which were set by [43].

- In the low-resource setting, we train LISA-LR by leveraging the Train sets of all the languages in the Multilingual Amazon dataset along with the Sentiraama Train Set. We then test the system on the Sentiraama Test set.

- In the no-resource setting, LISA-NR only utilizes the Train set of all the languages in the Multilingual Amazon dataset and test the system on the Sentiraama Test set.

Note: The Naive Bayes and SVM accuracies presented in the table differ from the ones presented by [43]. We attribute this to the difference in the train/test splits and the lack pre-processing guidelines which makes it hard to adequately replicate their results. The experimental results are given in Table 4.5

<table>
<thead>
<tr>
<th></th>
<th>Books</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>55</td>
<td>51.851</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>65</td>
<td>75.9</td>
</tr>
<tr>
<td>LISA-LR</td>
<td>72.5</td>
<td>85.185</td>
</tr>
<tr>
<td>LISA-NR</td>
<td>57.5</td>
<td>57.407</td>
</tr>
<tr>
<td>LISA-NoLD</td>
<td>67.5</td>
<td>68.51</td>
</tr>
</tbody>
</table>

Table 4.5 Results on the Sentiraama Dataset. The numbers denote binary classification accuracies.

The MUSE embeddings that were learnt were of size 300. The LSTM hidden states were of size 200. All the Dense Layers in the Sentiment Analyzer and the Language Discriminator were of size 200. We used the Adam optimizer [65] in all our experiments.
4.7 Analysis and Conclusion

Analysis: The results on the Multilingual Amazon Review Text Classification dataset proves our hypothesis that our model learns language invariant features that can be generalized across languages. The empirical results in Table 4.4 shows that our model outperforms pre-existing state-of-the-art methods on this dataset. While our experiments on the Sentiraama dataset proves that our model can be applied in a real-world setting to enhance sentiment retrieval in a truly low resource language. The ablation experiments (LISA-NoLD vs LISA-LR) show that between language pairs that have similar syntactic structure (example: English, French and German), LISA-LR performs much better than LISA-NoLD. This shows the the performance gains over prior work are not just due to the use of MUSE embeddings. Rather, they are attributed to the adversarial training of the Language Discriminator and the Sentiment classifier that extracts language agnostic sentiment features from the MUSE semantic space. But for Japanese (which is dissimilar with respect to other languages in the corpus), the results show that LISA-LR does not have a significant boots over LISA-NoLD. This is because our language adversarial training will retain only features that are invariant across all four languages, which is restrictive such that the information learnt will be too sparse to be useful. Finally, the poor performance of LISA-NR shows that our approach cannot be used for Zero-Shot learning but will achieve state-of-the-art performance in the presence of limited amounts of data in the low-resource setting.

Conclusions: In this chapter, we presented the LISA model which focuses on exploiting language invariant features for multilingual sentiment analysis without any form of cross-lingual supervision. We back our claims by conducting a wide range of experiments over the Multilingual Amazon Review Text Classification dataset and the Sentiraama dataset which is a real-world low resource dataset. We show that our model outperforms not only the existing cross-lingually unsupervised methods but also methods that rely on strong cross-lingual supervision. Additionally, our model sets the new state-of-the-art accuracies for the Sentiraama corpus.
Chapter 5

Learning Language Invariant Features for Open Domain Event Detection

“With limited training data, a more constrained model tends to perform better.”

— Christopher Manning

The contents of this chapter provides a primer on the task of open domain event detection and describes in detail the various components of the Multi-Lingual Sequence Tagger (M-LiST) architecture and its variants. We also explore different experimental set-ups to perform cross-lingual event detection for low-resource languages. As some of the contents of this Chapter is similar in structure to that of Chapter 4, we omit some details that we feel are redundant to enhance readability.

5.1 Introduction

The detection and extraction of events in text is an important Natural Language Processing and Information Retrieval task that has been successfully applied in many domains and in many languages. A myriad of events ranging from natural disasters, stock market variations, spread of pathogens to political events are reported on a daily basis. As a result of this seemingly ever-expanding creation of online content in the form of news articles, user blogs, messages, manuscripts and many others, it is vital to develop models that are capable of finding and extracting such events automatically, thereby saving valuable time and effort.

Detecting events in written text is a vital component for many applications and tools such as Question Answering systems, Dialog systems, Text Summarization systems and in Knowledge Graph creation. In view of its importance, automatic event detection has gained widespread attention among the NLP community with workshops, conferences and challenges (such as the TempEval1, TempEval2, TempEval3 challenges) conducted frequently.
The complexity of detecting events in text is compounded by the fact that there is no fixed definition on what an ”event” is. The dictionary definition of an event is ”a thing that happens or takes place”. This definition might be intuitive for humans to understand but for a computer, it might not be as informative. Therefore, throughout history, a lot of effort was invested into understanding and defining what events are. From a philosophical point of view, events were understood as relations between entities or objects that were influenced by time. The work by Kuhn [69] describes events as ”temporal intervals during which certain statements hold true”. Entities/objects play a key role in this definition of an event, without the concept of a event is meaningless [20]. This definition poses several challenges because the relations between different entities are temporal in nature and are usually not mentioned explicitly in text.

From the perspective of linguistics, various studies such as the work presented by Davidson [31] argued that events can be defined from the semantics of natural language from which we can infer causation and metaphysical determinism. Causation referring to the causal link between events and metaphysical determinism referring to the notion that events can be determined by other events preceding it. The pioneering studies of Mourelatos [90], Vendler [116] and the ”Algebra of Events” [8] by Bach introduced the broader notion of eventualities belonging to one of three classes : States, Processes and Events. Bach claimed that any phenomenon occurring in a particular duration can be classified into the three classes based on Durativity [89] and Telicity [100].

One of the earliest works that tackled the problem of event detection from a computational perspective was in the MUC-7 (Seventh edition of the Message Understanding Conference) where event detection was treated as a slot filling problem [82]. In this setting, events were defined as slots that links an action to its involved participants, the time at which the action occurred and the location. The Automatic Content Extraction (ACE 2005) program [33] was motivated by the MUC program that preceded it and address similar problems. But unlike the MUC program where the task was a slot filling problem, the ACE program tries to directly identify named entities, the relation and events. It provided annotation guidelines that defined an event as a specific occurrence that involved multiple participants. ACE was further enhanced with the Rich-ERE [109] corpora and through the Text Analysis Conference Knowledge Base Population (TAC-KBP) workshops 1.

Recent approaches to automatic event detection treat it as a sequence labelling problem where the input is a sequence of words and the goal is to classify the words into different categories that determine if the word belongs to an event or not. For example, according to the IOB annotation guidelines which takes into account the fact that some events can be expressed as multi-word phrases, there are three categories that a word can belong to:

1https://www.ldc.upenn.edu/collaborations/current-projects/tac-kbp
Automatic event detection in NLP can be broadly classified into two main categories:

1. **Open** domain event detection

2. **Closed** domain event detection

The term “domain” refers to different scenarios or settings that accounts for different overlapping subsets of events; such as, “movie domain” or “medical domain”. Although there exists numerous studies dedicated to the annotation and automatic extraction of events from texts, most of these studies are conducted in a specific domain with its own definition on what an event is. The task of finding events in text that belong to a particular domain, such as, monitoring seismic events\(^\text{[6]}\) or finding popular events from Twitter\(^\text{[5]}\) or finding events in sports\(^\text{[13]}\), is termed as "Closed domain event detection".

These task-specific definitions of events and its extraction often lead to constrained systems that perform poorly in other context and are impossible to employ in other domains. Conversely, "Open domain event detection" refers to the task of identifying and extracting all events in text regardless of the domain the event belongs to. In an open domain setting, we require a standardized, universal and linguistically motivated definition of an event such as the one that is provided by TimeML (described in the following subsection). We note that in this Chapter we focus exclusively on the task of open domain event detection.

### 5.1.1 TimeML

TimeML \(^\text{[101]}\) was one of the most prominent contributions for open domain event detection and temporal event linking which allowed defining events with finer granularity. TimeML (Time MarkUp Language) is a specification language for event and temporal expressions in natural language which was developed in the context of the AQUAINT\(^2\) program. TimeML was first introduced in 2002 as part of the Time and Event Recognition for Question Answering Systems (TERQAS) workshop in which the goal was to answer temporal-based questions regarding events in news articles. It was further extended by the Richer Event Description \(^\text{[91]}\) annotations in 2016. A revised and interoperable version of TimeML, termed ISO-TimeML \(^\text{[56]}\), was introduced in 2010.

\(^2\)https://ciir.cs.umass.edu/aquaint
TimeML presents a standard universal annotation guideline for defining events and temporal expression in written text. Although the TimeML project is similar in goal to the ACE project, the major difference is that TimeML is an open domain annotation project, whereas ACE is a closed domain annotation project. According to the TimeML guidelines, an event is described in a more generic fashion as “a cover term for situations that happen or occur” and a predicate as “describing states or circumstances in which something obtains or holds true”. These guidelines for annotation provides analyses of text at a word level which allows single words or multi-word phrases to be annotated as events. TimeML allows words to be marked with the following tags:

- \(TLINK\) : This tag denotes a temporal link that captures the temporal relations between events and temporal expressions.
- \(ALINK\) : This tag denotes an aspectual link that captures the aspectual relations between events.
- \(SLINK\) : This tag denotes a subordination link that captures the modal relations between events.
- \(TIMEX3\) : This tag denotes temporal expressions and their normalized values such as times or dates.
- \(SIGNAL\) : This tag is used to indicate the relation between temporal objects.
- \(EVENT\) : This tag denotes event expressions and their attributes (time, aspect, modality and polarity) and are classified into one of seven classes:

1. ASPECTUAL
2. I_ACTION
3. ISTATE
4. PERCEPTION
5. REPORTING
6. STATE
7. OCCURRENCE
5.1.2 TempEval Framework

<table>
<thead>
<tr>
<th>Language</th>
<th>Tokens</th>
<th>Event Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>53000</td>
<td>5688</td>
</tr>
<tr>
<td>Spanish</td>
<td>68000</td>
<td>12385</td>
</tr>
<tr>
<td>French</td>
<td>13000</td>
<td>2100</td>
</tr>
</tbody>
</table>

*Table 5.1 TempEval2 Dataset Statistics*

TempEval is a series of challenges that provides participants with datasets in multiple languages which were annotated with the TimeML tags. It presents an evaluation framework that allows direct comparisons between different event detection systems. The definition of an event in all the TempEval challenges follows the ISO-TimeML annotation guidelines. Through the years, there has been three editions of the TempEval challenges:

- **TempEval-1** [117]: The first edition of the TempEval challenges was focused on only extracting the relations between entities from English texts.

- **TempEval-2** [118]: The second edition of the TempEval challenges provided more precise definition for the tasks as compared to its previous edition. In the TempEval-2 challenge, the participants were provided with multilingual texts in Spanish and French along with the English texts from TempEval-1. The task was centered on extracting events, temporal expressions and temporal relations.

- **TempEval-3** [115]: The most recent of the TempEval challenges, TempEval-3 presented tasks that were centered on extracting events, temporal expressions and temporal relations. But unlike the TempEval-2 challenge, only Spanish and English texts were provided. Furthermore, it presented the additional task of evaluating the impact of the presence of automatically annotated training data along with the manual annotations.

In this chapter, we mainly focus on extracting events (verbs and nouns) from the dataset provided in the TempEval-2 challenge. Due to its multi-lingual nature, where annotated data is present in English, French and Spanish, makes it ideal to evaluate our proposed approach. The data is annotated in the IOB format where 'B' denotes the first token of an event, 'I' denotes all the other tokens present inside an event and O represents the tokens which are not a part of an event. The data statistics are presented in Table 5.1.
5.1.3 Event Detection: Low-resource Setting

Recent advances in deep neural architectures have facilitated impressive performances for sequence labelling tasks like automatic event detection [6]. The most common techniques usually employ a Bi-directional Recurrent Neural Networks (Bi-LSTM) to capture the temporal information between the words and a Conditional Random Field (CRF) layer for labelling [81, 70]. As with all deep neural networks, they are heavily reliant on large amounts of annotated datasets. The creation of such datasets is both a time and labour intensive task. Therefore, a majority of languages suffer from a dearth of annotated datasets which makes deep neural methods the wrong choice for them.

Therefore, prior research in low-resource event detection have resorted to unsupervised techniques or weakly-supervised methods that are heavily reliant on manual hand-crafted features. Most of these methods [53, 58, 74, 75, 76] attempt to develop an event detection system with the use of lexical (POS tags) features, syntactic features (dependency tags) or external knowledge (Wordnet features) [1] that are often built using language-specific resources or with the help of off-the-shelf NLP tool-kits such as the spaCy\(^3\) dependency parser or the Stanford Dependency\(^4\) parser. Although such approaches were shown to perform reasonably well, the lexical or syntactic features used are dependent on the language which makes these methods difficult to scale to other languages. Additionally, the process of engineering these feature is time consuming, labor intensive and prone to human error as it requires expert knowledge in that particular language.

Therefore, we conclude that a desirable approach should be able to learn features automatically from labelled data from other resource-rich languages that are language-independent so that it could be easily adapted to different languages. We ideally want to learn features that, in a way, captures the universal concept of ”event” irrespective of whatever language they occur in.

5.1.4 Proposed Approach: Overview

Driven by this motivation, the work presented in this Chapter tries to address some of the problems in low-resource event detection by introducing the Multi-lingual Sequence Tagger (M-LiST) architecture. The proposed architecture attempts to learn language-invariant features that are used for sequence labeling by training on labelled data from multiple high-resource (source) languages. This model trained end-to-end model can then be used for making predictions on sentences from a low-resource (target) language without the use of any language specific features or cross-lingual resources.

\(^3\)https://spacy.io/
\(^4\)https://nlp.stanford.edu/software/stanford-dependencies.shtml
Similar to the LISA architecture described in Chapter 4, we formulate the task as a **multi-lingual transfer learning** (MLTL) language adaptation problem [136] where we attempt to learn language-agnostic event features via adversarial training on labelled documents \{s_1, s_2...s_n\} from multiple resource-rich (source) languages to improve the performance on documents \{t_1, t_2...t_m\} from a low-resource (target) language. MLTL methods usually rely on parallel corpora [111, 35] or bilingual dictionaries [132, 40] to compensate for the difference in vocabularies of different languages. As stated before, these resources are not easy to develop. M-LiST alleviates the reliance on these cross-lingual resources by learning monolingual word embeddings from \{s_1, s_2...s_n\} \bigcup \{t_1, t_2...t_m\} and projecting them to a shared multilingual semantic space in a cross-lingually unsupervised fashion.

We employ an Bi-LSTM network to learn latent features (z) from this multilingual space along with character-level embeddings learnt by a Convolutional Neural Network (CNN), which is then used by a Sequence Labeller (S_L) to predict the event tags (B, I, O) of a word \(w_i\) in a document \(d \in \{s_1, s_2...s_n\} \bigcup \{t_1, t_2...t_m\}\). Concurrently, a language classifier (C_L) is trained to predict the language of document \(d\) based on \(z\). During the adversarial training we try to minimize the binary cross-entropy loss of \(C_L\), while at the same time we maximize the cross-entropy loss of \(C_L\). This results in a setting where the bi-LSTM learns to produce latent features (z) that predicts the event tags of document \(d\) correctly independent of the language of document \(d\). We hypothesize that in this setting, the latent features (z) trained would contain features that are language-agnostic which can then be leveraged for predicting the event tags in a low-resource language.

### 5.2 Multilingual Word Representation

The use of multilingual word representations for words in multiple languages alleviates the dependence on cross-lingual resources to train the M-LiST model. We employed the MUSE approach described in Chapter 3 and follows the same methodology as described in Section 4.3 to learn multilingual word embeddings for all the words in the TempEval2 dataset. We note that we used the pre-trained MUSE embeddings for English, French and Spanish that are hosted on GitHub\(^5\) and we did not need to train them like we did in Chapter 4 for the LISA architecture.

### 5.3 M-LiST Architecture

The input to the M-LiST model is a document \(d_i\) that is made up of a sequence of words \(w_1, w_2, \ldots w_k\). Each word \(w_i\) has associated with it a language label \(l_i \in L\) where \(L = \{l_1, l_2, \ldots l_p\}\) is the set of all language labels used in training. Furthermore, each word \(w_i\) has associated with it a event label \(t_i \in \{B, I, O\}\) which signifies if the word \(w_i\) is the **Beginning** of an event, **Inside** an event or **Outside** an event respectively.

---

\(^5\)https://github.com/facebookresearch/MUSE
Figure 5.1 The Multilingual Sequence Tagger (M-LiST) Architecture
For each word \( w_i \) we obtain its MUSE embeddings (from section 5.2) which we concatenate with character-level embeddings learnt by a CNN to obtain a sequence of \( n \)-dimensional word embeddings \( e_1, e_2, \ldots e_k \) where \( e_i \in \mathbb{R}^n \). The following subsections describes in detail the individual components of the M-LiST architecture. Figure 5.1 shows the overall architecture of the proposed model.

5.3.1 Feature Encoder (\( H \))

The purpose of the Feature Encoder (\( H \)) is to processes the sequence of word + character embeddings \( (e_1, e_2, \ldots e_k) \) using a Bi-directional Long Short Term Memory (Bi-LSTM) which sequentially, at each time step, outputs a hidden state \( h_i \) for every input embedding \( e_i \), such that:

\[
h_i = \text{LSTM}(e_i, h_{i-1}) \in \mathbb{R}^m
\]

We experimented with an LSTM, Bi-LSTM and GRU as our Feature Encoder and chose the Bi-LSTM results to report in section 5.5 as it produced the best performance. All the hidden states are then passed through a Language Discriminator (\( C_L \)) and a Sequence Labeller (\( S_L \)).

5.3.2 Language Discriminator (\( C_L \))

Similar to the Language Discriminator used in the LISA model, the goal of the Language Discriminator (\( C_L \)) in M-LiST is to predict the language label \( l_i \) based on \( h_i \). In other words, \( C_L \) tries to predict the language from which a particular word \( w_i \) comes from. Note that the classification is done at the word level in M-LiST, whereas it is done at the sentence level in the LISA model.

The \( C_L \) comprises of a Gradient Reversal Layer (\( GRL_\lambda \)), followed by two Dense Layers which uses the LeakyReLU activation function and an output Softmax Layer that applies the softmax function over all the languages used in training. During backpropagation, \( GRL_\lambda \) multiplies the gradients by a factor of \(-\lambda\) and during the forward pass it acts as the identity function. \( \lambda \) is hyperparameter in the network.
5.3.3 Sequence Labeller ($S_L$)

The Sequence Labeller ($S_L$) is made up of a Dense Layer followed by a Softmax Layer. Its goal is to assign an event label to each word $w_i$ using $h_i$. The softmax function ensures that the outputs of the Dense Layer are normalized between zero and one and that they sum to one at each time-step. Therefore, they can be interpreted as the posterior probabilities of the event tag given word.

![Figure 5.3 M-LiST : Sequence Labeller](image)

5.4 Training Set-Up

Inspired by recent works [47, 44, 14], we train the M-LiST model using adversarial training on a set of labeled documents $D = \{d_1, d_2, \ldots, d_n\}$ which belong to multiple languages. The aim of the M-LiST model is to predict the event tag ($t_i$) for a given word ($w_i$) independent of the language label ($l_i$).

We formulate the learning objective in a way that minimizes the event tag classification loss from $S_L$ and maximizes the language classification loss from $C_L$. As a result, the M-LiST model tries to jointly optimize the below functions:

$$
\arg\min_{H,S_L} f_1(S_L(H(w_i)), t_i) \quad (5.1)
$$

$$
\arg\max_{C_L} f_2(C_L(H(w_i)), l_i) \quad (5.2)
$$

Where $f_1$ denotes the cross entropy loss corresponding to the event label classification (B, I, O) produced by the sequence labeller $S_L$ and $f_2$ denotes the cross entropy loss function corresponding to the $L$ labels (English, French, Spanish) produced by the language classifier $C_L$ and is formulated as:

$$
\sum_{i=1}^{L} y_i \log(x_i) \quad (5.3)
$$
This results in a setting where the $C_L$ tries to predict $l_i$ based on a given $\mathcal{H}(r_i)$ and the encoder $\mathcal{H}$ tries to 'fool' the $C_L$ by learning to create $(w_i)$ that is minimally influenced by the language label $l_i$ while at the same time, is maximally influenced by the $S_L$ to predict the event tag $t_i$ correctly. With the incorporation of a Gradient Reversal Layer $GRL_{\lambda}$ [44] between $\mathcal{H}$ and $C_L$, the network learns to correctly predict the event tags $t_i$ and at the same time, learns to incorrectly predict the language labels $l_i$. By using $GRL_{\lambda}$, the optimization functions (equations 5.1 and 5.2) can be simplified as:

$$\arg\min_{\mathcal{H}, S_L, C_L} f_1(S_L(\mathcal{H}(w_i)), t_i) + f_2(C_L(GRL_{\lambda}(\mathcal{H}(w_i))), l_i)$$

(5.4)

5.5 Experiments and Results

In this section we present an extensive set of experiments conducted on the TempEval2 Dataset. We evaluate the M-LiST architecture in the different settings that are described below:

1. **M-LiST-L**: We evaluate the performance of the M-LiST architecture in the low-resource setting (termed **M-LiST-L**) by training it on the Train sets from multiple source languages and the limited Train set in the target language and then testing on the Test set of the target language. We perform experiments in this setting to evaluate the performance of the M-LiST model for few-shot learning tasks.

2. **M-LiST-U**: In the no-resource setting, we assume that the training data is not available for the target language. We train this variant of the M-LiST model (termed **M-LiST-U**) on the Train sets of the source languages and evaluate the model on the target language Test set. We perform experiments in this setting to evaluate the performance of the M-LiST model for zero-shot learning tasks.

3. **M-LiST-D**: To show the effectiveness of the Language Discriminator ($C_L$), we conducted ablation experiments in the low-resource setting where we remove $C_L$ from the M-LiST model. In this variant of the M-LiST architecture (termed **M-LiST-D**), the Sequence Labeller ($S_L$) only depends on the MUSE + Char embeddings to learn $h(w_i)$. We perform the ablation experiments with **M-LiST-D** to show that the Language Discriminator ($C_L$) is a vital component of the M-LiST architecture.
Table 5.2 Performance of M-LiST on the TempEval2 Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>English</th>
<th></th>
<th></th>
<th>French</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>$F_1$</td>
<td>P</td>
<td>R</td>
<td>$F_1$</td>
</tr>
<tr>
<td>Current Best</td>
<td>86.0</td>
<td>86.0</td>
<td>86.0</td>
<td>84.7</td>
<td>76.4</td>
<td>80.3</td>
</tr>
<tr>
<td>M-LiST-L</td>
<td><strong>86.2</strong></td>
<td><strong>86.7</strong></td>
<td><strong>86.4</strong></td>
<td>84.4</td>
<td><strong>87.5</strong></td>
<td><strong>85.9</strong></td>
</tr>
<tr>
<td>M-LiST-U</td>
<td>45.5</td>
<td>70.0</td>
<td>55.1</td>
<td>76.8</td>
<td>59.8</td>
<td>67.5</td>
</tr>
<tr>
<td>M-LiST-D</td>
<td>83.2</td>
<td>83.6</td>
<td>83.4</td>
<td>84.9</td>
<td>86.3</td>
<td>85.5</td>
</tr>
<tr>
<td>M-LiST-UD</td>
<td>48.9</td>
<td>62.1</td>
<td>54.7</td>
<td>74.4</td>
<td>59.7</td>
<td>66.2</td>
</tr>
</tbody>
</table>

4. **M-LiST-UD**: This variant of the M-LiST architecture is a combination of M-LiST-U and M-LiST-D which is trained without the language discriminator ($C_L$) and without relying on any training data from the target language.

We compare our results against prior state-of-the-art methods on this dataset presented below:

1. **Spanish**: The previous state-of-the-art model for open domain event detection in Spanish for the TempEval-2 dataset was established by [128]. They used a 70-30 train-test split. They experimented with a Conditional Random Field and Support Vector Machine using syntactic and morphological features like POS-tag, word lemma, mood, tense, etc, that were extracted from the Freeling [7] tagger. They also used manual hand-crafted features that were associated with word structure like using capital letters in the words. They used a window size of [-2 to 2] to train their CRF. They reported an average F1 score of 80.3.

2. **English and French**: The previous state-of-the-art model for open domain event detection in English and French for the TempEval-2 dataset was established by [5]. They experiment with Conditional Random Fields, Decision Trees and K-Nearest Neighbors using syntactic features such as word lemmas and POS-tags. For English, an additional feature was used to denote if a particular word belonged to one of the eight classes of synsets that were associated with open-domain events. Similarly, in French they used an additional feature was used for each word which indicated whether it belongs to the VerbAction [113] and The Alternative Noun Lexicon [15] lexicons or not. They reported an average F1 score of 83 for French and 86 for English.
We used a CNN with 30 filters with a filter-width of size 3 to learn character embeddings of size 300. The MUSE embeddings are also of size 300. The Bi-LSTM hidden states are of size 100 in both the forward and backward propagation and concatenate to a final hidden state of size 300 which is used by the Sequence Labeller and the Language Discriminator. We used the Adam optimizer [65] in all our experiments.

5.6 Evaluation Metrics

To achieve consistency with prior research and to provide direct comparisons with them, we measure the performance of the M-LiST models on three standard metrics: Precision, Recall, and F-Measure (F1).

Precision denotes how precise or accurate the model is. It calculates out of those predicted positive, how many of them are actual positive and is given by:

\[
Precision = \frac{TP}{(TP + FP)}
\]  

(5.5)

Recall calculates the percentage of the Actual Positives our model captures out of those that the system was supposed to recognize:

\[
Recall = \frac{TP}{(TP + FN)}
\]  

(5.6)

F1-score or F-Measure is calculated as the weighted harmonic mean of Precision and Recall.

\[
F1 = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)}
\]  

(5.7)

5.7 Analysis and Conclusion

Analysis: The results of the M-LiST model on the TempEval2 dataset proves our hypothesis that our model learns language-invariant features that can be generalized across languages for open domain event detection. Our architecture allays the reliance on language-specific handcrafted lexical features that most prior methods required for event detection in text. The empirical results presented in Table 5.2 shows that our model outperforms pre-existing state-of-the-art methods for all the languages from the TempEval2 dataset. The ablation experiments that we conducted shows that the M-LiST-L model consistently outperforms the M-LiST-D model which highlights the importance of the Language Discriminator in the M-LiST architecture.
We also observed that the M-LiST model shares some of the disadvantages suffered by the LISA architecture. More concretely, we see from Table 5.2 that there is no significant performance increase as compared to prior work in English. This is in stark contrast to the F1 scores that we achieved for Spanish and French which is significantly higher than the ones reported in prior works. We attribute this to the fact that French and Spanish belongs to the same Romance branch of the Indo-European language family, hence, they are similar in syntactic structure. This reaffirms our inference that our proposed approach performs better when the source languages are similar in terms of their syntactic structure.

**Conclusions**: In this chapter, we presented the M-LiST model and its variants which focuses on exploiting language invariant features for multilingual open domain event detection without using any form of cross-lingual supervision or language-dependent features. We back our claims by conducting a wide range of experiments on the TempEval2 dataset, attaining state-of-the-art performance in the low-resource setting (M-LiST-L). The results of our experiments reaffirm our confidence in our approach aiming to learn patterns from multiple languages, that captures the concept of “events” that is common across languages, which can then be used for event detection in a low-resource language. Finally, the poor performance of M-LiST-U and M-LiST-UD shows that our approach cannot be used for Zero-Shot learning where we have no labelled data in the target language.
Chapter 6

Conclusions and Future Work

As described in Chapter 1, NLP research in low-resource languages are hindered by the lack of annotated data which impedes the development of NLP tools and services for that language. The work presented in this thesis provides an alternate approach to solve NLP tasks for low-resource languages such that the benefits of NLP can be enjoyed across all languages. In a nutshell, our work builds on prior research in the domains of multilingual word embeddings and cross-lingual transfer learning (described in Chapter 2) to formulate an approach that leverages annotated data in multiple resource-rich languages to learn features that are language-agnostic without using cross-lingual resources or any form of cross-lingual supervision. The learnt language-agnostic features can then be used to perform NLP tasks for low-resource languages.

First of all, we addressed the need for alleviating the dependence on cross-lingual resources like bilingual lexicons or parallel corpora which are commonly used for performing cross-lingual transfer learning between resource-rich languages and resource-poor languages. Chapter 3 dives deep into solving this problem with the use of the unsupervised MUSE approach. We bridge the language barrier that cross-lingual resources are used for by aligning the word embeddings of vocabularies of different language into a common multilingual MUSE embedding space. We note that for our experiments in sentiment analysis, we decided to train our own MUSE embeddings, rather than using the pre-trained models. We justify our decision due to the unavailability of pre-trained models for some languages such as Telugu and Japanese.

We then proceeded to test the viability of the proposed approach on a classification task. Consequently, we chose the task of sentiment analysis which provides us with a plethora of datasets and prior research and models that we compare our system with. Chapter 4 details the conception of the LISA architecture and its variants and the methodology it follows to find the sentiment polarity of written text in a low-resource language. The LISA system achieved state-of-the-art performance for all the languages in the Amazon Multilingual Text Classification dataset and the Sentiraama corpus which proved our hypothesis that our approach is capable of learning language-agnostic sentiment features from resource-rich languages that can be used to improve the performance in low-resource languages.
With the encouraging results of the LISA model for sentiment analysis, we proceeded to test the robustness of our approach on Open Domain Event Detection which is a sequence labelling task. Chapter 5 describes in detail the various components of the M-LiST architecture and its variants and the training methodology we followed to detect events in low-resource languages. Our end-to-end M-LiST model that was trained on multiple languages without the use of any language specific features or cross-lingual resources achieved state-of-the-art performance on the tempeval2 dataset. The results of the experiments presented in Chapter 5 reaffirm our confidence in our approach to learn patterns and features that are common across languages.

Although, the initial goal of our work was to develop a method that could perform well in the absence of any annotated data (for Zero-Shot learning tasks), our experiments in sentiment analysis and open domain event detection conclude that the model performs poorly in this scenario and we are disappointed to report that our approach will not perform well on unseen languages. Although, in the presence of minimal amounts of annotated data, where we treat it as a Few-shot learning task, we find that our architectures perform significantly better than most unsupervised approaches and even approaches that rely on strong cross-lingual supervision. Ablation experiments described in Chapter 4 and Chapter 5 confirms our hypothesis that the use of MUSE embeddings were not the sole cause of the performance gains reported and that the language-invariant features learnt by our networks play a crucial role.

We also note that the proposed approach is more suitable to languages that belong to the same language group. For example, our architectures perform better when the source languages share similar syntactic and semantic structures. Therefore, for the Japanese language in the Amazon Multilingual Text Classification dataset which is dissimilar with respect to other languages in the corpus, our results show that LISA-LR does not have a significant performance gains over LISA-NoLD. Which means that our language-agnostic features are not as important to the prediction of sentiment as compared to the information provided by the MUSE embeddings for the Japanese language. This is because our language adversarial training will only retain features that are invariant across all four languages, which in many cases, is too restrictive such that the information learnt will be too sparse to be useful. Hence, we can infer that the performance will degrade as we add more dissimilar source languages during training.

6.1 Future Work

Although the scope of this thesis is limited to sentiment analysis and open domain event detection, we plan to explore the applications of our language-agnostic architectures on other NLP tasks and also it’s ability to perform well in other low-resource languages. The impact of our architectures are not limited to NLP tasks as well. As of writing this thesis, we are conducting experiments to perform emotion-invariant language identification which is a speech processing task which we believe can be
solved with minimal alterations to the approach presented in this thesis. We also plan to experiment with other multilingual embedding spaces since we employed the MUSE approach exclusively in all our experiments which might not produce the most optimal performance. In all our experiments we had tested with LSTMs, Bi-LSTMs, GRUs and Bi-GRUs as our encoder. Future work in this direction should also include the use of other novel architectures such as the transformer architecture or the BERT network as the encoder in place of the RNN-based networks that we used in our experiments.

The ability of our architectures to perform well in the presence of limited amounts of training datapoints with no cross-lingual resources available, makes it a compelling choice to use in such settings. Although we believe we have left a lot of questions unanswered, we hope that the work presented in this thesis establishes a solid foundation that enables future research in this direction.
Related Publications


Bibliography


