Translation of Named Entities Between Sinhala and Tamil for Official Government Documents

Thayaparan Mokanarangan

(178089G)

Degree of Master of Science (Research)

Department of Computer Science And Engineering

University of Moratuwa
Sri Lanka

August 2018

Translation of Named Entities Between Sinhala and Tamil for Official Government Documents

Thayaparan Mokanarangan

(178089G)

Thesis submitted in partial fulfillment of the requirements for the
Degree of Master of Science (Research) in Computer Science and Engineering

Department of Computer Science And Engineering

University of Moratuwa
Sri Lanka

August 2018

Declaration

I, Thayaparan Mokanarangan, declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signed: _____

Date: _____

The above candidate has carried out research for the Masters Dissertation under my supervision.

Name of Supervisor: Dr. Surangika Ranathunga

Signature of supervisor: _____ Date: _____

Name of Supervisor: Dr. Uthayasanker Thayasivam

Signature of supervisor: _____ Date: _____

"It was awesome, but also.. it wasn't?"

Troy from Community

UNIVERSITY OF MORATUWA

Abstract

Faculty Of Engineering
Department of Computer Science And Engineering

Master of Science

by Thayaparan Mokanarangan
(178089G)

Analyzing existing machine translation approaches for Sinhala-Tamil official government documents have revealed the shortcomings when translating named entities. The diverse nature of the domain coupled with the lack of resources and morphological complexity are the key reasons for this problem. Our research focuses on translating named entities for official government documents between Tamil and Sinhala. In this research, we focus on identifying and translating named entities to improve the translation performance. We present a novel tag set specific to official government documents and also propose a graph-based semi-supervised approach that works better than state-of-the-art approaches for low-resource settings. We employed this approach to build a large annotated corpus in a cost-effective manner from a smaller amount of seed data and was able to build an annotated corpus of over 200K words each for Tamil and Sinhala. We also implemented a deep-learning approach for Named Entity Recognizer that gave the best output for a completed corpus. Since the deep-learning approach was a generic solution for sequential tagging, we also employed it to build a Part-of-Speech tagger that outperforms existing systems. The University of Moratuwa already has a system for translating official government documents called *SiTa*. Finally, we incorporated the aforementioned models to build a module that translated named entities and integrated it to *SiTa*. We empirically show that our modules improve over the baseline for Tamil $\rightarrow$ Sinhala and Sinhala $\rightarrow$ Tamil translation tasks by upto 0.5 and 1.4 BLEU scores, respectively.

Keywords: Machine Translation. Named Entity Recognition, Graph-Based Semi-Supervised Learning, Deep Learning, Named Entity Translation

Acknowledgements

I would never have been able to finish my dissertation without the guidance, support and encouragement of numerous people including my mentors, my friends, colleagues and support from my family. At the end of my thesis I would like to thank all those people who made this thesis possible and an unforgettable experience for me.

First and foremost, I would like to express my sincere gratitude to my supervisors Dr. Surangika Ranthunga and Dr. Uthayasanker Thayasivam, for the continuous support given for the success of this research both in unseen and unconcealed ways. This would not have been a success without your tremendous mentorship and advice from the beginning. Your wide knowledge and logical way of thinking have been of great source of inspiration for me. You have always extended his helping hands in solving research problems. The in-depth discussions, scholarly supervision and constructive suggestions received from you have broadened my knowledge. I strongly believe that without your guidance, the present work could have not reached this stage.

I wish to thank Prof. Gihan Dias and Prof. Sanath Jayasena for their supervision, advice, and guidance from the very early stage of this research as well as giving me extraordinary experiences through-out the work. This research was supported by the Department of Official Languages and the University of Moratuwa Senate Research Grant. I sincerely thank the colleagues from the Department of Official Languages for the support given.

I would like to thank Ms. Fathima Farhath, Ms. Nimasha Dilshani and Ms. Yashothara Shanmugarajah, who as good friends from my graduate studies, were always willing to help and give their best suggestions.

Thank you!

Contents

List of Figures

List of Tables

Abbreviations

NLP	**N**atural **L**angaguge **P**rocessing
NER	**N**amed **E**ntity **R**ecognition
NE	**N**amed **E**ntities
BLEU	**B**i-**L**ingual **E**valuation **U**nderstudy
MT	**M**achine **T**ranslation
CRF	**C**onditional **R**andom **F**ield
LSTM	**L**ong **S**hort **T**erm **M**emory
GSSL	**G**raph **B**ased **S**emi-**S**upervised **L**earning
CBOW	**C**ontinous **B**ag **O**f **W**ords
ME	**M**aximum **E**ntropy
HMM	**H**idden **M**arkov **M**odel
POS	**P**arts **O**f **S**peech
SMT	**S**tatistical **M**achine **T**ranslation
NMT	**N**eural **M**achine **T**ranslation

Chapter 1

Introduction

This thesis focuses on enhancing statistical machine translation for official government documents between Sinhala and Tamil by identifying and translating Named Entities (NEs).

Our research has been divided into two main parts: named entity recognition and translation. This chapter introduces the tasks, the motivation, research methodology and contributions made by this research.

1.1 Overview of Named Entity Recognition

Named Entity Recognition (NER) is the process of identifying named entities in natural language text. Typical named entities can be classified as Person, Organization and Location [1]. Figure 1.1 illustrates a simple example of named entity recognition on a sample text.

Introduced with the 6^{th} Message Understanding Conference, 1995 [2], NER has become an integral part of many natural language processing challenges including question answering [3, 4], information extraction [5] and opinion mining [6].

Earlier NER approaches centered on using manually coded rules and mapping of dictionaries [7]. While giving better results for restricted domains, these approaches failed to detect complex named entities.

FIGURE 1.1: An example of NER application on an example text
Source: http://imanage.com/wp-content/uploads/2014/10/NER1.png

Hence, researchers seek to solve this by employing machine learning algorithms using large annotated corpus (a large and structured set of texts used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific language territory). These approaches are more robust, efficient and effective when compared to rule-based approaches. Hidden Markov Model (HMM) [8], Maximum Entropy (ME) [9] and Conditional Random Fields (CRF) [10] are popular machine learning approaches. Out of these approaches CRF has been the most successful [1].

However, recently most of machine learning has been engulfed by the performance of deep-learning methods. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [11]. The current state-of-the-art approach [12] employs a deep learning method comprised of Bi-directional Long Short Term Memory (LSTM) CRF and deep contextual vectors. Meanwhile researches for Sinhala and Tamil NER have been limited to traditional sequential tagging algorithms [13–16].

Despite its performance, deep-learning methods require a large amount of annotated data to produce the best results. Building an annotated corpus consumes resources and requires expert knowledge.

In such cases, where supervised data is scarce, it has been common to employ semi-supervised learning (SSL) techniques for many different Natural Language

Processing (NLP) tasks [17, 18]. In general, graph-based semi-supervised learning (GSSL) techniques have shown better performance than other SSL techniques [19]. But the key challenge on employing GSSL with regards to NER is to build a graph that is capable of capturing the contextual information.

1.2 Overview of Machine Translation

Machine Translation is the process of translation from one language to another with the aid of computer. Early translation techniques that focused on word-to-word substitutions between the two languages failed to produce accurate translations.

The short comings of this approach led to Rule Based Machine Translation (RBMT) that generates the output based on morphological, syntactic, and semantic analysis of both the source and the target language [20]. But RBMT proved to be difficult to incorporate rule interactions in big systems, ambiguity, and idiomatic expressions.

In recent times, MT turned to the use of corpus to ensure translation of whole phrases of text to their closest counterparts in the target language. For nearly two decades, Statistical Machine Translation (SMT) has been the widely used method. It generates translation based on on statistical models with parameters derived from the analysis of bilingual and monolingual corpora [21]. But SMT systems are domain dependent and fails to translate between language with significant grammatical different like Romance and Dravidian languages.

The current research trend in MT is Neural Machine Translation (NMT). NMT builds a single neural network that is trained using deep-learning techniques to maximize the translation accuracy. The models proposed recently for neural machine translation often belong to a family of encoder-decoders and consist of

an encoder that encodes a source sentence into a set of annotation vectors from which the decoder generates translation [22].

Despite the promises of NMT, it requires a very large parallel corpus to train. This requirement is a setback language pairs that lack the luxury of having a large parallel corpus. For instance, NMT [23] for Sinhala-Tamil language has given significantly low output when compared to a SMT system [24]. The SMT system proposed by Farhath et al. [24] called *SiTa* was developed by the University of Moratuwa for translating official government documents. It employs the Moses translation system [25] for its translation purposes.

1.3 Motivation

The current *SiTa* system's performance suffers because of its inability to translate named entities. Unlike general domain documents, official government documents have diverse set of named entities. Analyzing *SiTa* had revealed its shortcomings in translating varying person names like சுமாலி புத்திகா/සුමාලි බුද්ධිකා and town locations like சம்மாந்துறை/සම්මන්තුරේ. This is because with every document there is a new name or location to identify and translate. Though there has been research conducted for NER in Sinhala and Tamil, they are primarily concerned with general domain. There is no corpus available for official government documents. Couple this with the lack of performance of the existing systems means, if used in translation process, it would reduce the translation quality.

1.4 Research Objectives

1. Identify a named entity tag set for the domain of official government documents

2. Implement a machine learning technique that performs well in low-resourced settings and use that to annotate data, which will then be cleaned by human annotators.

3. Apply state-of-the art deep-learning approaches for NER, once a sufficiently large data set is created

4. Design a translation and transliteration mechanism for identified named entities

5. Integrate the above-mentioned approaches to the existing SMT system.

1.5 Contributions

1. Identified a unique tag set specific to official government documents

2. Designed a novel graph-based sequential tagging approach that outperformed the state-of-the-art approaches for low-resource settings

3. Designed a deep-learning based named entity recognizer and Part-of-Speech (POS) tagger for Tamil and Sinhala

4. Developed an unsupervised approach to identify morphological transformations and build a named entity translation module

5. Improved the translation quality of the existing SMT system

1.6 Articles

- Presented our paper "Graph based semi-supervised learning for Tamil POS tagging" in LREC-2018 conducted at Miyazaki, Japan
  - H-Index : 43
  - World Rank 5^{th} on Computational Linguistics

1.7 Organization of the Thesis

This thesis shows how named entities can be identified and translated in Sinhala-Tamil official government texts. The organization of this thesis is as follows.

- **Chapter 2** discusses the literature related to identifying and translating named entities. The summary and conclusion drawn with regard to each aspect of our literature is also briefly explained. The methodology was derived from these conclusions.
- **Chapter 3** explains the method employed to achieve the research objective. It details out each aspect of our research and how it pertains to the goal.
- **Chapter 4** briefly shows the implementation decisions taken to realize the methodology in code.
- **Chapter 5** lists the experiments and results carried out with respect with to the methodology and the research. An in-depth analysis derived from each results is also listed.
- **Chapter 6** concludes our research with the empirical information obtained from the experiments.
- **Chapter 7** lays out the plan for future work that could be accomplished as an extension of this research.

Chapter 2

Literature Review

This chapter details the literature related to identify and translate named entities. It has been divided into 5 main sections.

Section 2.1 introduces the theory and rationale behind Named Entity Recognition. It discusses common challenges with regards to designing a NER system and also introduces some contemporary corpus related to this research.

Section 2.2 discusses the research conducted so far related to identifying named entities. While the research has been extensive, one of the recurring challenges was that most of them yielded poor results in low resource settings. Hence we branched out to find semi-supervised approaches that are proven to work well in low resource settings.

Section 2.3 delves into a special branch of the semi-supervised approach, Graph-based Semi Supervised (GSSL) approach. In this section, we briefly discuss different approaches conducted with regards to NLP tasks and sequential tagging in particular.

One of the key factors we identified in *Section 2.3* is the need to represent words effectively in machine learning models. *Section 2.4* details the existing distributional semantic models proven to effectively capture syntactic and semantic information of words.

Finally, in *Section 2.5* we discuss the literature that pertains to Machine Translation (MT). The section lists out two of the most popular MT approaches - Statistical Machine Translation (SMT) and Neural Machine Translation (NMT). We also discuss existing approaches for Sinhala-Tamil translation and translating named entities. The section then concludes with the conclusion derived regarding MT for Sinhala-Tamil NEs.

2.1 Named Entity Recognition

Named entity recognition (NER) is the problem of locating and categorizing important nouns and proper nouns in a text [26]. NER plays a vital role in NLP tasks. It helps in extracting information, identifying relationships between different information and decision making. NER has been used as a key feature for Question Answering [3, 4], Information Extraction [27] and Opinion Mining [6].

2.1.1 Challenges in NER

Following are some of the general challenges with regards to NER.

- Generative in Nature

  Day-to-day named entities grow in size. For example, when a baby is born, if the parents keep a unique name for that baby, the name is a new addition to the named entity set. Similar logic applies also for organization and date. This makes the named entity Recognition task more complex [28].

- Variations of Named Entities

  The same named entity can be represented using various different forms. For example Prof. Malith Fernando can also be represented as Dr. Malith Fernando, Dr. Fernando, Malith Fernando and A.P.J. Malith Fernando [28].

- Ambiguity Named Entity Types

  There is a possibility that more than one entity type can have the same names. For example, "May" can represent the month, but can also represent a person [28].

- Morphological Complexity

  This is an issue that occurs with morphologically complex languages like Tamil or Sinhala. Consider the example மோகனிடம்/ අමිලට and மோகனால் / අමිලගෙන්, both represent the same person with different inflection.

- Lack of Resources

  The best approaches require labeled data [10]. The higher the amount of data, the better the performance. This again is a challenge for low resource languages like Tamil and Sinhala. For instance, the CoNLL NER corpus [1] for English has over 300K tokens, whereas the FIRE Corpus for Tamil only has 80K tokens [29].

2.1.2 Datasets

This section details some of the prominent datasets used in contemporary researches.

MUC-6 Corpus NER came into focus during the 6^{th} Message Understanding Conference, 1995 [2]. As a part of the proceedings MUC-6 corpus was built with the following tag set:

- ENAMEX - organizations, persons, locations
- TIMEX - times (dates, times)
- NUMEX - quantities (monetary values, percentages)

CoNLL Dataset Introduced as a part of the CoNLL 2003 shared task [1], CoNLL dataset is widely considered as a benchmark for testing new approaches. It has a very simple tag set with 4 different tags: Person, Location, Organization and Miscellaneous. The follow listing shows a small snippet of the corpus. Here each columns denote the token, POS, Chunking and NER tags respectively. This corpus consists of over 300K tokens.

```
...
Derbyshire NNP I-NP I-ORG
and CC I-NP O
Surrey NNP I-NP I-ORG
all DT O O
...
```

Listing 2.1: CoNLL 2003 Dataset Snippet

FIRE Corpus Designed by the Forum for Information Retrieval (FIRE) [29], this provides tagged data for major Indian languages including Tamil, Malayalam, Telungu and Hindi. The data is arranged in a similar manner to the CoNLL corpus but with a different tag set. Unlike the CoNLL corpus, this has 87 different tags including GOVERNMENT, VEG, INDIVIDUAL, NATION and PLACE. It consists of 80K tokens.

SiTa Parallel Corpus Used by Farhath et al. [24] and Tennage et al. [23] to design their SMT and NMT Sinhala-Tamil translation system respectively. This is a parallel corpus housing over 25000 sentences primarily focusing on official government documents. Its currently been used to train the *SiTa* system.

2.2 Existing Approaches for Named Entity Recognition

Recognizing previously unknown entities is an essential part of the NER system. While early studies mostly relied on handcrafted rules, most recent ones employ

deep-learning approaches. The rest of this subsection details the literature related to building NER systems.

2.2.1 Rule-based Approaches

Rule-based systems are based on manually coded rules and manually compiled corpora. The output is produced based on matching of the rules and dictionary entries. These systems generally consist of a set of patterns using grammatical (e.g. part of speech), syntactic (e.g. word precedence) and orthographic features (e.g. capitalization) in combination with dictionaries. They give better results for restricted domains, and are capable of detecting complex entities that learning models have difficulty with. Also, they are ideal for the languages and domains for which there are limited linguistics resources available [7].

Appelt et al. [30]'s FASTUS was one of the earliest systems for NER that relied on handcrafted regular expressions. Later, Grishman [31] improved upon the approach by incorporating dictionary and Part of Speech (POS) tags.

However, these systems are less portable and robust. Furthermore, the maintenance of the rules costs a lot as even a slight change in the data requires an intense manual work. These types of approaches are often domain and language specific. Consequently, they do not necessarily adapt well to new domains and languages [32]. Hence, researches turned to machine learning approaches that would scale up to new data and identify hidden information between words.

2.2.2 Machine Learning Approaches

In Machine Learning based approaches, the identification problem is converted into a classification problem and statistical classification models are used to solve it. In this type of approach, the systems look for patterns and relationships within the text to make a model using statistical models and machine learning

algorithms. Since these approaches function on the statistical details, they are more domain and language independent than that of rule-based ones.

NER is a sequential tagging problem. In sequential tagging problems, the label of a word is predominantly determined by its context. Thus, syntactic relationships between word tokens play a major role. For example, consider the NEs "Central Bank spokesman" and "The Central African Republic". Here, the word 'Central' is used as part of both an Organization and Location, depending on the context. Thus, without referring to the context, the exact NER tag of the word cannot be determined.

Hence, researchers employed structured prediction algorithms that encode the sequence information. *Hidden Markov Model* (HMM) was the earliest approach used to solve NER. HMM is a graphical model, where the conditional dependence between random variables can be expressed by a graph, that allows expressing the conditional probability distributions based on a limited history (the Markov property) [8].

Bikel et al. [33]'s *IdentiFinder* was the earliest model that attempted to solve NER using HMM. It uses the locality of phenomena that indicates names in text. For example, if a word is preceded by "Mr." or "Mrs.", then, it is bound to be a person's name.

Within each of the word context regions, they use the statistical bi-gram language model for computing the likelihood of words occurring with each named entity. This statistical model computes the likelihood of sequence of words by employing a Markov Chain. The Viterbi algorithm [34] was used to efficiently search the entire word space for possible named entity assignments. Bikel et al. [33] had reported an accuracy of 94.69% and 90% for a mixed case English and mixed case Spanish MUC-6 data.

Later, Zhou and Su [35] modified this approach by incorporating the internal feature of the words such as capitalization, digitalization, semantic feature of

important triggers and gazetteers. They had reported an accuracy increase of 2% and 4% than Bikel et al. [33]'s approach.

Though Zhou and Su [35] used a wide variety of features, their model performed poorly for the CoNLL 2002 data. This was largely due to the machine learning method rather than the feature set [9]. One of the main challenges with HMM is that it fails to incorporate a diverse set of overlapping features [9]. On the other hand, the *Maximum Entropy* (ME) model handles a diverse set of overlapping features easily.

Unlike HMM, ME models discriminate. It directly learns the weight of features for classification based on the training data. Objective of the ME model is to generalize as much as possible for training data by maximizing the entropy for data. Curran and Clark [9] employed the maximum entropy model to design a language independent named entity tagger. They had reported an F1-Score of 84.9 for CoNLL-2003 English and 68.4 for CoNLL-2003 German data.

But ME models have a weakness called the *label bias problem*. It fails to identify the global context of transition properties [36]. In some cases the role of the second token will be lost in distinguishing the entity type. To counter this problem researchers used *Conditional Random Fields* (CRF).

Introduced by Lafferty et al. [36] as a statistical modeling tool for pattern recognition and machine learning using structured prediction [10], until recently CRF are considered to be the most successful classification for NER [8]. The concept of CRF is based on ME. The key difference is that while ME models use per-state exponential models, CRF uses a single exponential model for an entire label sequence label [37]. This allows CRF to classify the whole sequence at once unlike ME, thus helping to sidestep the *label bias problem*. Implemented by Stanford University, Stanford NER is one of the widely used CRF based NER Tagger (See Section 2.2.10.1).

While all the above-mentioned approaches are based on structured prediction algorithms, McNamee and Mayfield [38] proposed *Support Vector Machines* (SVM) as a potential candidate for NER. In contrast to the above-mentioned approaches, SVM are incapable of directly capturing sequence. To overcome this issue, McNamee and Mayfield [38] came up with 258 different features that would potentially capture the sequential information. Following are some of the features:

- Token length equal to 0-9
- Token length between 10 and 15, or greater than 15
- Whether the token was preceded by comma, a full stop, a question mark, an exclamation mark or other punctuation
- Whether the token is succeeded by comma, a full stop, a question mark, an exclamation mark or other punctuation

For CoNLL 2003 [1] they had reported a F1-Score of 57.8 and 59.2 for Spanish and Dutch datasets respectively. Thus, gaining parity with contemporary machine learning approaches.

Each approach has its own strengths. Researchers tried to combine the strength of every approach and create a hybrid approach called the *Vote based classifier ensemble technique* [39]. The key idea behind the classifier ensemble technique is that this is often much more precise than other individual classifiers that make them up. Generalization accuracy of ensemble is high. It mainly depends on the diversity of each individual classifier as well as on their individual performance [39]. Hence, appropriate classifier selection for constructing an ensemble system remains a difficult problem. In addition, not all classifiers are good to detect all types of output classes. For example, some classifiers are more precise at detecting organization names whereas some are more precise at detecting person names. Thus, in a voted system, a particular classifier should only be allowed to vote for that output class for which it performs well.

Saha and Ekbal [39] proposed a model where there was one classifier for each kind of named entity. For instance, there were different classifiers for person, organization, and location. The resulting system was an ensemble system of classifiers. This approach had yielded good results for Bengali, Hindi and Telungu [39].

In an ensemble system, it is necessary to find out either the set of classes for which a classifier is most suitable to vote or to search for the appropriate weights of votes for all the classes in each classifier.

2.2.3 Semi-supervised Approaches

When labeled data is scarce, its common to employ semi-supervised approaches [40, 41]. Semi-supervised algorithms used both labeled and unlabeled corpus during classification.

Carreras et al. [42] proposed one of the earliest semi-supervised approaches that used AdaBoost [43] with confidence rated predictions as the learning algorithm for the classifier. Three binary classifiers corresponded to Beginning (B), Inside (I) and Outside (O) of NE Tag. Orthographic and semantic features were used over boosting algorithm combining several fixed-depth decision trees. Although simple in nature, the approach relied heavily on hand-crafted language features.

On the other hand, Graph based Semi-Supervised Learning (GSSL) is a branch of semi-supervised learning that uses graphs to solve NLP problems. Researchers have shown that these methods are generally more effective than traditional semi-supervised approaches [44]. Talukdar and Pereira [41] had introduced a language independent (GSSL) approach for NER that uses WordNet to build graphs and subsequently employ label propagation algorithms for classification. This approach only considers the surface form of words and is incapable of capturing the local context information necessary for a sequential tagging problem like NER. GSSL and GSSL for sequential tagging are discussed in detail at Section 2.3.

2.2.4 Unsupervised Approaches

In contrast to supervised and semi-supervised approaches, unsupervised approaches do not require any labeled data. The general approach in this learning is clustering. For example, these approaches tend to gather NEs from clustered groups based on the similarity of context. These methods typically rely on lexical resources (e.g. WordNet), lexical patterns and statistics computed on a large corpus [45].

Alfonseca and Manandhar [46] proposed an approach based on WordNet. This approach assigns a topic to each WorNet synset by listing words that frequently co-occur with it in the corpus. Then, based on the input word, the word context was compared to the type signatures and clustered under the same label.

Alfonseca and Manandhar [46]'s approach requires WordNet, which is a resource that is still lacking for many languages. Meanwhile, Shinyama and Sekine [47] used the observation that NEs often appeared synchronously in news articles when compared to the common nouns. They exploited this approach to cluster the NEs in an unsupervised manner.

Based on the same correlation observation, Etzioni et al. [48] proposed an experimental system called *KNOWITALL*. This approach uses Pointwise Mutual Information and Information Retrieval (PMI-IR) to assess the correlation between words and classify them. Though this approach has been promising, the precision of this system is as low as 0.57 [48].

The inability to define a custom tag set and the general lack of performance have been the major setbacks with unsupervised methods.

2.2.5 Cross Lingual Approaches

Cross lingual approaches attempt to transfer the learning from one language to the another. In this case, it transfers from a high-resource language like English to a low-resource language like Catalan or Galician so as to increase the accuracy of the latter system. This cross lingual approach is also aided with the vast amount of parallel data that can be extracted for free from Wikipedia.

Cotterell and Duh [49] proposed a solution, given a low-resource target language, they additionally offer large amounts of annotated data in languages that are genetically related to the target language. Using this approach, they have shown to increase accuracy for Indo-European (Catalan, Galician) and Austronesian (Tagalog, Cebuano) languages.

This approach was an extension of work done by Pan et al. [50], where the authors had carried out experiments for 282 languages. Provided a document in any of the given languages in Wikipedia, their framework was able to identify name mentions, assign a coarse or fine-grained type to each mention and link it to an English knowledge base. This approach performed well for Wikipedia text but under performed for non-Wikipedia text.

Both of these approaches are based on building a knowledge base and then transferring the knowledge. In contrast, Yang et al. [51] built a Long Short-Term memory based CRF system that encodes both character-level and word-level information across languages. They then used the leverage of training with languages that share same character-level morphology to increase the accuracy.

All these approaches have performed well within genetically similar language families but fail across different ones. For instance, Pan et al. [50]'s cross lingual approach yielded a F1-Score of 93.9 for Spanish but only yielded a F1-Score of 77.0 for Tamil. This under performance also aggravated by the questionable quality of the parallel text available in Wikipedia.

2.2.6 Deep-Learning Approaches

In recent years, deep-learning methods have proven to be more effective than traditional machine learning in various branches of Computer Science [11, 12, 52]. Deep-learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [11]. It has improved the results in speech recognition, object detection, machine translation and other complex machine learning related problems.

One of the first approaches with deep-learning for sequential tagging was proposed by Huang et al. [52]. They had proposed a bi-directional Long Short Term Memory (LSTM) with a CRF layer. In contrast to traditional machine learning algorithms, LSTM are capable of storing long-term dependencies thus connecting with previous learned information to the current task. This property has helped to effectively capture the long-term context of each word and yield state-of-the-art results for POS, Chunking and NER.

Later Lample et al. [53] extended this research by proposing a Convolutional Neural Network (CNN) to encode the characters and capture the morphological information. Similarly, Ma and Hovy [54] proposed an LSTM network to encode the characters instead of a CNN. Both of these approaches increased the F1-Score points by 2 for English [53, 54].

With a lot of hyper parameters to tune, Reimers and Gurevych [55] ran benchmark tests to identify which of them were optimal. Their research revealed that using CNN or LSTM for character encoding had little over bearing to the system performance. In contrast, usage of distributional semantic vectors like (See Section 2.4) GLoVe [56], FastText [57] or Word2Vec [58] produced varying results for different tasks.

Recently, Peters et al. [59] had proposed a deep contextual vector representation called ELMo (See Section 2.4.5). Unlike other existing neural word embedding,

this approach takes the context of word into account while producing the vector. Plugging this vector to the bi-directional LSTM CRF has produced the best results so far with 91.93 % for CoNLL 2003 NER task and 96.37% for CoNLL 2000 Chunking task. But, when they sampled 1% of the CoNLL 2003 data set, their performance dropped down to 71.01%.

2.2.7 Existing Approaches Used for Tamil and Sinhala NER

All of the above mentioned approaches centered on English or other Latin languages like German and Spanish. Both Sinhala and Tamil pose different challenges when compared to these languages. The rest of the section details the research carried out for these languages.

2.2.7.1 Tamil

All the approaches used for Tamil had employed sequential tagging algorithms. Vijayakrishna and Sobha [16] and Malarkodi et al. [60] had employed CRF, meanwhile Theivendiram et al. [13] had used Margin Infused Relaxation Algorithm (MIRA). The differentiating factors are the domain, tag set and features used.

Vijayakrishna and Sobha [16] and Malarkodi et al. [60] had used a hierarchical tag set of 106 different tags. While the previous one focused specifically on tourism domain, the latter was a general tagger. Vijayakrishna and Sobha [16] had used roots of words, Parts of Speech tags and word patterns as their features. On the other hand, Malarkodi et al. [60] had used morphological and POS features.

Theivendiram et al. [13] used a simpler tag set consisting of Individual, Place, Organization, Time and Count. In addition to the features proposed by Malarkodi et al. [60], they had also used suffix as features. They had reported the best F1-Score of 80.38 for Forum for Information Retrieval (FIRE) [29] NER Tamil corpus.

2.2.7.2 Sinhala

Similar to Tamil, all the research conducted with regard to NER centered on sequential tagging algorithms. Dahanayaka and Weerasinghe [15] and Manamini et al. [14] had experimented with CRF and ME for different feature, tag and corpus set.

Dahanayaka and Weerasinghe [15]'s feature set includes the word suffix and bi-gram features. They had built a binary classification corpus with 75K words and reported an F1-Score of 68.04. Manamini et al. [14] on the other hand had employed a richer feature set including length of word, frequency of word, POS tags and gazetteers. They had built a corpus of over 110K tokens with Location, Organization and Person as their tag set to report a F1-Score of 78.

2.2.8 Existing Approaches Used in Different South Asian Languages

South Asian languages also face the same challenges faced by Sinhala and Tamil. Table 2.1 details different techniques and features used with NER for some popular South Asian languages.

Most of the research conducted, especially with regards to Tamil and Sinhala, have used different tag sets and the corpus used with these approaches are not also publicly available. This makes it hard to come to a conclusion on which feature set works the best. But the advantage of CRF over other machine learning algorithms is evident. As noted by Dahanayaka and Weerasinghe [15] ambiguities with words, free word order, agglutintative nature and lack of resources are some of the common problems faced when designing NER for Sinhala and Tamil.

Implementation	Languages	Method	Features
Li and McCallum [61]	Hindi	CRF	Word text (prefix, suffix), gazetteer, Features at the current, previous and next sequence positions
Saha et al. [62]	Hindi	ME	Statistical linguistic feature set, gazetteer, context patterns
Patel et al. [63]	Hindi, Marathi	Inductive Logic Programming(ILP)	Manually extracted rules from tagged corpus
Saha and Ekbal [39]	Bengali, Hindi, Telungu	ME, CRF and SVM (Weighted ensemble model)	Word text, context information
Gali et al. [64]	Bengali, Hindi, Telungu, Oriya	CRF	Language specific heuristics
Nayan et al. [65]	Hindi	Rule Based	Matching similar phonetic with different languages

TABLE 2.1: Different approaches for NER in Indian languages

2.2.9 Features in NER

Various features have been used for NER. Based on the context of the features, they are categorized as local or global. If only a small neighborhood is considered, it will be categorized as local, whereas if the whole document or corpus is considered, then it is categorized as global features [32]. In addition to these local and global features, resources have also been used. The rest of the section details some of the common features and resources used for NER.

2.2.9.1 Local Features

Orthographic features : They are based on the appearance of the word, which mainly focusing on the characteristics of the characters that make the word (E.g. Begins with upper case, mixed cases, all in upper case, word with mixed digits). This is a simple and language independent feature but not effective for many languages where capitalization is not applicable.

Affixes : The begin pattern or ending pattern of the word is considered. This feature is also language independent, but the pattern of the prefix or suffix has to be selected with careful consideration of the language and domain.

Word : The characteristic of the word in multiple instances can be considered as another feature (e.g. the capitalization of the first letter of a word at the beginning of the sentence as well as in the middle).

N-grams : Two types of N-grams are used in NER. They are character N-gram and word N-gram. Character N-grams capture the inner structure of the word. Affix can be considered as a special case of character N-gram. Word N-gram are used to identify the word sequence of NEs.

Part of speech and morphology : Part of speech helps to identify the nouns and adjectives that are most likely to be NE candidates and tags like verbs and prepositions that are less likely to be the candidate. Morphology helps to detect the constructive words that in turn can help to improve the detection of NEs.

Patterns : Pattern is a string of category characters. Rule-based systems are mostly pattern based. In machine learning, patterns can be used as a feature by using some automatic pattern extracting mechanisms.

Word Embedding : Using word embedding models with machine learning approaches has yielded the state-of-the-art results for NER [12]. Their ability to capture syntactic and semantic information is discussed in detail at Section 2.4.

2.2.9.2 Global Features

Previous appearance : If a text is marked as NE previously, then another instance of the same text has the most likelihood to be a NE. This is mostly used in the classification in defining the class to which it belongs to, than in recognition.

Meta Information : Meta information can be used directly to retrieve NEs, such as mail headers. But these features are heavily domain dependent [32, 45].

2.2.9.3 Resources

Gazetteers : Gazetteers are a list of NEs. For a word to be a candidate, it should exactly match with an element existing within the list. But some normalization procedures (i.e. stemming, lemmatization) may be required before matching.

Trigger Words : These are a list of words that are not NEs but they are often found in the neighborhoods of the NEs. E.g. the word 'Honorable' can trigger a Person NE. These rules either can either be handcrafted or learned through a corpus.

2.2.10 Available Platforms and Toolkits

Software reuse helps to efficiently circumvent pre-existing trivial challenges like reading data sets, setting up configuration and defining features. It also helps in organizing the implementation. In that aspect, following are some famous toolkits and platforms available for NER.

2.2.10.1 Stanford NER

Stanford NER [66] is a Java Implementation of CRF based tagger, implemented by Stanford University. This is licensed under GNU General Public License. The pre-existing model for English is designed to identify *Person*, *Location* and *Organization*. Following are some of the features used.

- **Word Features:** Current word, previous word, next word, left three words, and all words within a window
- **Orthographic features:** John $\rightarrow Xxxx$
- **Prefixes and Suffixes:** Brown $\rightarrow < B, < Br, < Br, ...own >, wn >, n >$

2.2.10.2 GATE Named Entity Recognizer

Gate [67] is a project of the University of Sheffield. Unlike Stanford NER, this project supports many tasks related to NLP. Gate provides the infrastructure for developing and deploying applications for Natural Language Processing. It can identify *Date*, *Location*, *Money*, *Organization*, *Percentage*, *Person*, and *Time*.

2.2.10.3 Natural language Toolkit (NLTK)

This toolkit contains features that are needed for natural language processing including named entity recognition. It is purely based on Python and there are many corpus available for the public usage. 90% of the corpus is for the English language and 10% of the remaining corpus is for other languages [68]. This toolkit has many features including concordance (mostly occurring pair words), tagged corpus and the techniques to extract the information from corpus and conditional frequency distributions of corpus. These types of features are very helpful for the development of a proper NER.

2.2.11 Evaluation Measures

For the the first named entity recognition system in MUC-6 [69], both the precision and recall were reported as outputs. Here, precision is the percentage of correctly named entities found by the system. Recall is the percentage of named

entities in the corpus that have been extracted by the system. The entity is considered as correct only if it has the exact same span and type.

But since the CoNLL 2003 Shared Task [1], F1-measure has been the evaluation benchmark for recent researches [12, 52]. The performance of the system was measured with $F_{\beta=1}$ rate as shown in Equation 2.1. In order to benchmark our results with contemporary researches, we have adopted the F1-measure as the evaluation measure for our NER.

$$F_\beta = \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * precision + recall} \tag{2.1}$$

2.2.12 Summary

Relevant literature has revealed that the deep-learning approach of bi directional LSTM sequential tagging with character embedding [12] to be the ideal candidate for building a NER. Its language and task independent approach coupled with the high output makes it a viable candidate for other languages. This approach can also be extended to other sequential tagging tasks like POS and Chunking. The only hindrance in this approach is that it requires a high amount of data to perform optimally.

Either cross lingual or semi-supervised approaches can solve the low resource problem. However, cross lingual approaches require high-resource languages from the same language family. There is no high-resource language available from the Indo-Aryan or Dravidian language family to which Sinhala and Tamil belong respectively. Hence, the natural candidate be a semi-supervised approach.

As discussed, even within semi-supervised approaches, GSSL has proven to be more effective and simple to design. The next section details the literature with regards to GSSL with NLP.

2.3 Graph Based Semi-Supervised Learning (GSSL)

Graph theory and Natural Language Processing are well-studied disciplines, but commonly perceived as distinct with different algorithms and applications [70]. But recent research has shown that these disciplines are connected and graph-theoretical approaches can be employed to find efficient solutions for NLP problems. In many NLP problems, entities are connected by a range of relations in many NLP problems and a graph is a natural way to capture the relationship between the entities. GSSL has been used in word sense disambiguation, entity disambiguation, thesaurus construction, textual entailment and semantic classification.

GSSL builds graphs connecting labeled and unlabeled data points, and perform classification by propagating the labels. The graph is constructed to reflect our prior knowledge about the domain. The intuition is that similar data points have similar labels. We let the hidden/observed labels be random variables on the nodes of this graph. Labels are injected to unlabeled nodes from labeled nodes. Graphs provide a uniform representation for heterogeneous data and are easily parallelizable [71].

One of the challenges of a graph-based approach is building the graph that reflects the relationship between entities. Depending on the task, the nodes and edges may represent a variety of language related units and links. Different NLP tasks have approached this challenge in different ways.

For example, consider text normalization in social media languages. The social media language is constantly evolving with new phonetic substitutions and slang words. Text normalization helps increase the performance of processes that rely on these data, like machine translation. Hassan and Menezes [72] had proposed a method that propagated the correct word forms to alternative spellings. The graph for this approach was built from social media text extracted and a large

clean corpus where each nodes are either denoted by correct word forms or slangs. Here the correct words are connected to the unlabeled noise words. Label propagation was then employed to identify the correct versions of unlabeled nodes.

Another example is text summarization, where unlike the previous example, just denoting each nodes by words is not appropriate. We will have to represent sentences or phrases with each node. Zhu et al. [73] constructed a graph of sentences linked by edges whose weight combines the term similarity and objective orientation similarity. In this approach, a set of leaders is iteratively extracted from a graph communities of a sentence. After generating the community leaders using link propagation, they are selected to generate the summary.

Taking it to the next level, consider the challenge of discourse identification. Discourse is not merely based on sentences but also on the context of the conversation. Hence representing each node with sentences will not be enough. Elsner and Charniak [74] predicted the probabilities for pair of utterance as belonging to the same conversation thread or not based on lexical, timing and discourse-based features. They then constructed a graph with each node representing the utterances and the edges representing the probability score between the nodes.

These examples depict how different challenges have attempted different graph building methods. As stated earlier, named entity recognition is a sequential tagging approach, so it is paramount to build a graph that is capable of capturing context information. The following section details the research carried with regards to graph based approaches for sequential tagging.

2.3.1 Graph-based Approach for Sequential Tagging

Early work on using GSSL for sequential tagging problems relied on word-based graph representations. Talukdar and Pereira [41] had constructed a word graph using WordNet to perform NER. In this approach, vertices are noted as surface-level word forms and each relationship in WordNet is represented as an edge.

Although simple and straightforward, this approach fails to capture the syntactic information essential for sequential classification tasks.

In contrast, Subramanya et al. [40] represented each vertex using a vector of pointwise mutual information (PMI) values, computed using the n-gram and each of the features that occur with tokens of that n-gram. The cosine distance between these PMI vectors of a pair of vertices are used as edge weights between those vertices. These PMI vectors are capable of capturing local context information. However, they note that the vectors used in this approach are sparse and high dimensional.

Extending on Subramanya et al. [40]'s work, Das and Petrov [75] designed unsupervised POS taggers for languages that have no labeled training data. They constructed a graph based on the same PMI features introduced by Subramanya et al. [40], and used graph-based label propagation for cross-lingual knowledge transfer. This solution was based on the observation that despite the language differences, words in different languages share similar relationships in local context.

In their research on graph-based posterior regularization for semi-supervised structured prediction, He et al. [76] claimed that using Subramanya et al. [40]'s features to build graphs leads to the unrelated matching of trigrams to match. Instead, they proposed a different set of features to build PMI-based graphs. However, this also suffers from sparsity.

Recently, Demirel [77] had proposed an approach to solve POS tagging where every word in a corpus is connected to a graph and each node is denoted by a word-embedding vector. They capture the word ordering information by connecting each word to the next and previous word in the corpus. This graph is then directly fed into a neural network model called graph convolutional network (GCN) for classification.

2.3.2 Summary

The literature of GSSL has revealed to us that though it has been successfully used extensively in other natural language challenges, the approaches relative to sequential tagging has been few and far. One of the key challenges with using graphs for sequential tagging is to capture the local context of words. While different researchers have tried with different approaches, we believe using distributional semantic models will yield a better performance. The following section discusses the possible semantic models that can be used.

2.4 Distributional Semantic Models - DSM

Due to its efficiency and simplicity, Distributional Semantic Models (DSM) have become a fundamental part of natural language processing [12]. DSM are the process of representing words as vectors.

One of the simplest models to represent a word as vector is *one hot vector representation*. In this representation, if the size of vocabulary is denoted by $|V|$ then the word represented will have a dimension of $|V|$ where only the index of the word is active and the rest is set to zero.

For example:

Cat : [0000.....1....00]

Dog : [0000.....1.....00]

Though simple and easy to implement, there is no correlation between two words and a large vocabulary leading to high sparsity. Distributed semantic models help solve the sparsity problem and captures the semantic and syntactic information that is essential for language processing tasks.

DSM can be divided into two categories [78]:

1. Count Models - Traditional models like Positive Pointwise mutual information and Local Mutual Information [79] that use co-occurrence counts to create vector models

2. Predict Models - Neural language models like Word2Vec [80], FastText [57] that predict the context based on the word

The following section describes some of the prominent distributional semantic models.

2.4.1 Pointwise Mutual Information (PMI) Vector

The traditional way to represent words by vectors is to build a high dimensional sparse matrix M, where each row is represented by a word w in the vocabulary, and each column is a potential context c. Each matrix cell, M_{ij} shows the association between word, w_i and context, c_j [81]. Introduced by Church and Hanks [82], pointwise mutual information (PMI) is a way of measuring the association. Equation 2.2 defines the association between w and c.

$$PMI(w, c) = \log \frac{(w, c).|D|}{(w).(c)} \tag{2.2}$$

In this equation:

- D is the number of total word-context pairs
- (w, c) is the number of occurrences both the context and word has appeared together
- (w) is the total number of times that word has occurred
- (c) is the total number of times the context has appeared

For word context pairs that will never co-occur, PMI value would be $-\infty$. To tackle this issue positive pointwise mutual information (PPMI) is used where 0 replaces all negative values. Equation 2.3 shows how PPMI is calculated.

$$PPMI(w, c) = max(PMI(w, c), 0) \tag{2.3}$$

The main drawback of this approach is the sparsity of vectors. With increasing corpus size, the vector size continues to increase.

2.4.2 Word2Vec

Introduced by Mikolov et al. [83], Word2Vec is one of the most popular choices for pretraining projection matrix $W \in R^{d \times |V|}$ where d is the embedding dimension with vocabulary, V [84]. As illustrated in Figure 2.1 [85], two models were defined - Continuous Bag of Words (CBOW) and skip gram. The objective of the skip-gram model is to maximize the likelihood of the prediction of contextual words given for the center word. Meanwhile CBOW predicts the center word based on the context words.

Out of the two models, research has revealed that skip-gram models are found to capture semantic information effectively than CBOW [55].

2.4.3 FastText

Proposed by Bojanowski et al. [86], FastText is another version continuous word representation trained on large unlabeled corpus. Similar to Word2Vec, this approach also proposed skip-gram and CBOW models. But in contrast to Word2Vec, FastText relies on subword information where the vector representation is represented as the sum of character n-grams. The goal of this approach is not only to

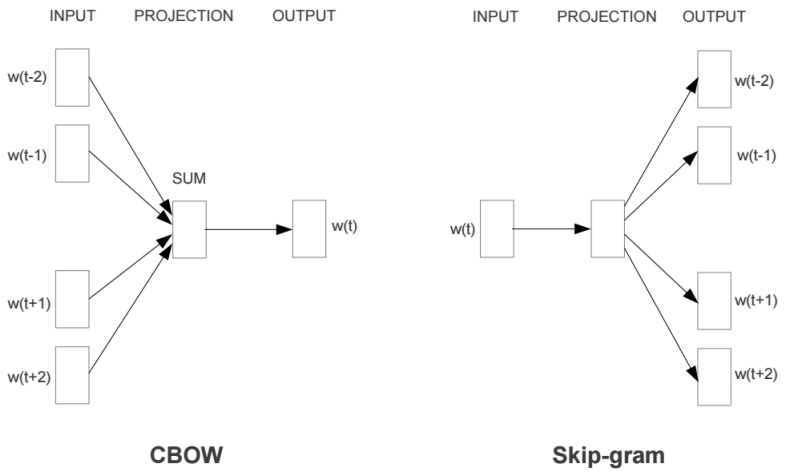

FIGURE 2.1: CBOW Vs Skip-gram models
Source: https://raw.githubusercontent.com/rohan-varma/paper-analysis/master/word2vec-papers/models.png

capture the word context but also the morphological information present in each word.

2.4.4 Wang2Vec

Another simple modification of Word2Vec, Wang2Vec [84] generates models that are more suited towards syntax related problems. While the original Word2Vec model is insensitive to word order, Wang2Vec captures them to induce syntax-sensitive word embedding. This has proven to show improvements in Part-Of-Speech (POS) tagging and dependency parsing [84].

2.4.5 ELMo

Recently introduced by Peters et al. [59], the ELMo model is a very powerful model that is capable of capturing the past and future contexts of words. Unlike other models, this generates different vectors for the same word depending on the

context. Incorporating this for the sequential tagging problem has produced the state-of-the-art results for NER and Chunking in English [59].

2.4.6 Summary

Baroni et al. [78]'s research has claimed that neural predict models better capture semantic information when compared to predict models. Levy et al. [81] have claimed that with proper pre-processing and hyper parameters, there are no discerning advantages between them. But recently Mikolov et al. [87] have claimed that neural language based models such as FastText yields a higher gain than statistical based model like GloVe.

Though there are contrasting claims with regards to count and predict models, it is generally agreed that count models are high in sparsity. Thus, rendering it hard to use with a high amount of data.

Continuous vector models capture syntactic and semantic information that can be exploited with the deep learning and graph based approaches. For instance, with these approaches, all the person names will be clustered in a separate space. We can exploit this aspect in our research to build a better sequential tagging model.

2.5 Machine Translation

The final phase of our research is to translate the identified words and increase the performance of the existing translation system. This section briefly introduces two main translation approaches: Statistical Machine Translation (SMT) and Neural Machine Translation (NMT). It also details existing translation approaches with regard to Sinhala-Tamil and named entities.

2.5.1 Statistical Machine Translation - SMT

The first task when building a machine translation system is to collect pairs of source sentences and their corresponding translations. (X_n, Y_n) will be used to represent a pair of source and corresponding translation, respectively. D is the data set with N pairs. With the training data, D, in hand, it is possible to score a model by looking at how well the model works on the testing data. The score, which is called the log-likelihood of the model, is the average of the log-likelihood of the model on each pair of sentences. With the probabilistic interpretation of the machine translation model, the log- likelihood of the model on each pair is simply how high a log-probability the model assigns to the pair. θ is the set of parameters that defines the model.

The overall score of the model on the training data is defined by Equation 2.4.

$$L(\theta, D) = \sum logp(\frac{y^n}{(x^n, \theta)}) \tag{2.4}$$

The aim is to ensure high score of log-likelihood, L, if the score is low it means that the model is wasting its probability mass on wrong translations. Hence, it is paramount to find a configuration of a model that maximizes the scoring. This approach is called *maximum likelihood estimator*. The core of Statistical Machine Translation (SMT) is a log-linear model, where the logarithm of the true $p(y|x)$ is approximated with a linear combination of many features. A large part of the research comes down to finding a good set of feature functions.

SMT automatically maps sentences in one human language into another. The translation model can be formulated as shown in Equation 2.5 [88] where the goal is to find the most likely target sequence, t^*, for some source sequence, s. Here, s stands for the *source* language and t stands for the target language.

$$t^* = argmax_t P(s|t)P(t) \tag{2.5}$$

This approach encompasses three main aspects:

- **Translation Model ($P(s|t)$):** Specifies the set of possible probabilities for some target sequence
- **Language Model ($P(t)$):** Models the fluency of the proposed target sequence.
- **Argmax operation:** Searching through the word space for possible target translations. This is called *decoding*.

The SMT system uses a log-linear model as shown in Equation 2.6 [89].

$$e^* = argmax_e(\sum_i f_i(e, f)\lambda_i) \tag{2.6}$$

Here feature functions, $f_i(e, f)$, captures the aspect of translation and each of them has a weight of λ_i. The weights are scaling factors that are adjusted with respect to loss functions, which evaluates the translation quality. Typically, this function is calculated in terms of the BLEU score (refer Section 2.5.3.1) .

2.5.1.1 Moses

Moses is a statistical machine translation system that allows to automatically train translation models for any language pair using a collection of translated texts (parallel corpus) [25]. Once a model is trained, an efficient search algorithm quickly finds the highest probability translation among the exponential number of choices.

Features

1. Offers phrase-based and tree-based translations
2. Offers factored translation models, which enable the integration of linguistic and other information at word level.
3. Moses allows the decoding of confusion networks and word lattices enabling integration with ambiguous upstream tools like speech recognizers or morphological analyzers

The translation model is trained with a parallel corpus and language model. The language model in turn is typically trained on corpus akin to the domain of the translation task.

2.5.2 Neural Machine Translation - NMT

Neural machine translation does not rely on pre-designed feature functions. The goal of NMT is to design a fully trainable model of which every component is tuned based on training corpora to maximize its translation performance. Figure 2.2 shows the high-level architecture diagram of NMT.

Given a source sequence X = $\{x_1, x_2, x_3, x_4...x_T\}$ of word indices, the NMT model computes the conditional probability of Y = $\{y_1, y_2, y_3...y_{T'}\}$. While traditional phrase based translation systems use a pipeline of sub components tuned separately, NMT builds and trains a single, large neural network that reads a sentence and outputs a correct translation [22]. Unlike SMT, NMT learns directly, in an end-to-end fashion [90].

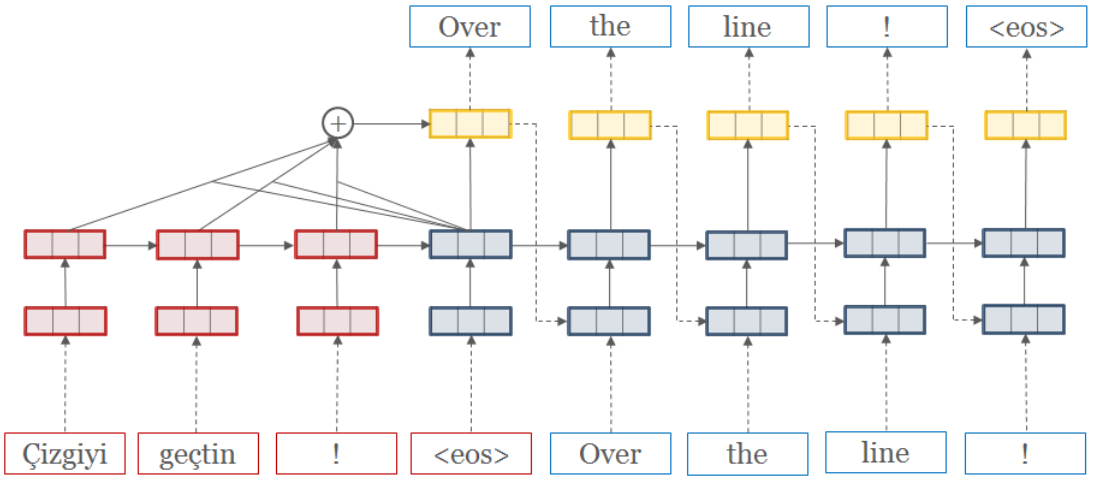

FIGURE 2.2: Neural Machine Translation
Source: http://opennmt.github.io/simple-attn.png

2.5.2.1 Encoder-Decoder Model

The source sentence is first encoded into a fixed-length vector by an encoder neural network. Then the translation is derived from the encoded vector using a decoder neural network. This is the encoder-decoder model, which encompasses the decoder and encoder for a language pair jointly trained to maximize the probability of a correct translation [22]. The key aspect is of this model is that this can work with varying length of parallel text because of its ability to encode the text into a fixed-length vector.

2.5.3 Evaluation

2.5.3.1 BLEU Score

BLEU (bilingual evaluation understudy) is an algorithm for evaluating the quality of text that has been machine-translated from one natural language to another [91]. Quality is the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is"– this is the central idea behind BLEU. BLEU was one

of the first metrics to claim a high correlation with human judgments of quality, and remains one of the most popular automated and inexpensive metrics.

Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. Intelligibility or grammatical correctness is not considered.

BLEU is designed to approximate human judgment at a corpus level, and performs badly if used to evaluate the quality of individual sentences. BLEU's output is always a number between 0 and 100. This value indicates how similar the candidate text is to the reference text, with values closer to 100 representing more similar texts. Few human translations will attain a score of 1, since this would indicate that the candidate is identical to one of the reference translations. For this reason, it is not necessary to attain a score of 1 as there are more opportunities to match and adding additional reference translations will increase the BLEU score.

BLEU has frequently been reported as correlating well with human judgment, and remains a benchmark for the assessment of any new evaluation metric [92]. There are however many criticisms that have been voiced. It has been noted that although in principle it is capable of evaluating translations of any language, BLEU cannot in its present form, deal with languages lacking word boundaries.

2.5.3.2 NIST Score

The NIST metric [93] is another method for evaluating the quality of the translated text. It is based on BLEU score with some alterations. Where BLEU simply calculates n-gram precision adding equal weight to each one, NIST also calculates how informative a n-gram is. When a correct n-gram is found, the rarer that n-gram is, the more weight it will be given.

NIST also differs from BLEU in its calculation of the brevity penalty insofar as small variations in translation length do not impact the overall score as much. NIST metric uses a heavier weight for rare words thus leading to results that are hard to interpret for low-resource languages.

2.5.4 Existing Machine Translation Systems for Tamil-to-Sinhala Translation

Similar to NER, the key challenge for MT in Sinhala and Tamil is the lack of resources. Most of the approaches barring one exception by Tennage et al. [23] have used Moses SMT system in combination with GIZA++ translation model [94]. Each research has been attempted with different language models and approaches. Table 2.2 details these SMT approaches.

Implementation	Language Model	Approach	BLEU Score	
			Tam-Sin	Sin-Tam
Weerasinghe [95]	CMU-Cambridge Toolkit [96]	4064 manually aligned sentences	0.0618	0.1362
Sripirakas et al. [97]	SRILM [98]	5697 parallelly aligned sentences constricted to Parliament order papers obtained from UCSC-LTRL [99]. Built the Sinhala and Tamil language models with 6566 and 75051 sentences respectively	0.4277	0.5599

Pushpananda et al. [100]	SRILM [98]	25500 sentences parallelly aligned sentences. Built the Sinhala and Tamil language models with 407,578 and 850,000 sentences respectively	10	13
Rajpirathap et al. [101]	IRLSTM [98]	5000 parallelly aligned phrases. Built the Sinhala and Tamil language models with 6550 and 6104 sentences	0.5957	0.6693
Pushpananda et al. [102]	SRILM [98]	Extended their own work [101] by incorporating an unsupervised morphological analyzer using the Morfessor algorithm [103]. Morfessor algorithm was used to find morpheme-like units of the source and target languages in order to build the translation and language models. This algorithm has better segmentation accuracy and handles Out Of Vocabulary (OOV) words	12	15

TABLE 2.2: SMT and GIZA++ based approaches for Sinhala-Tamil Translation

As is evident by the Table 2.2, Sinhala-Tamil translations consistently produce better results when compared with Tamil-Sinhala. This is because of the phonetics and morphological complexity of Sinhala over Tamil. It is also evident from the table that with better data, the performance of the system has gone up.

As pointed out by Weerasinghe [95]'s research, the two main challenges with MT in Sinhala-Tamil are:

- Limited size of corpora highlighted by the high perplexity of Sinhala and Tamil language models
- Long-distance "movement" of mutually translated words and phrases captured in current translation models

2.5.4.1 *SiTa* SMT system

The *SiTa* system currently runs the model designed by Farhath et al. [24]. Similar to previous approaches, this also uses Moses system with GIZA++ translation model [94]. They have employed SRILM [98] to train the language model. It uses in-domain and out-domain data to increase the translation efficiency. In-domain contains official letters from government department and out domain data collected from other government sources such as annual reports, parliament order papers, circulars, and establishment codes. 22,073 parallely aligned sentences. Their approach have yielded a BLEU score of 37.01 for Tamil→Sinhala and 46.14 for Sinhala→Tamil translation. We discuss it separately since we are using this as our baseline approach.

While the above-mentioned centered on SMT, Tennage et al. [23] had designed a NMT system for official language documents. They used the same dataset used by Farhath et al. [24] and reported a BLEU score of 12.75 and 7.5 for Sinhala→Tamil and Tamil→Sinhala translation, respectively.

2.5.5 Existing Approaches to Translate Named Entities

Named entity translation has so far been an amalgamation of MT and transliteration. One of the initial research conducted by Huang and Vogel [104]. They

had employed a NER in a bilingual parallel corpus independently to extract the NE list, align the NEs using a statistical alignment model and build a statistical translation model.

To alleviate the dependency of a parallel corpus, Rapp [105] proposed an approach to try and obtain NE's from non-parallel text. They came up with an approach comprised of co-occurence counting and vector similarity to identify NE translations.

Since most named entities can be transliterated, many works build these models based on the rule-based approach [106] or statistical approach [107, 108]. Rule-based approaches use linguistic rules to generate translations. This will be harder when transliterating from a less phonetically complex language towards a higher complex one.

Statistical based transliterations on the other hand select the most proba-++ble one based on the training data. Virga and Khudanpur [108] had proposed an approach that deconstructed both English and Chinese NEs into phonemes and trained the translation model on the aligned phonemes. While Virga and Khudanpur [108] had employed a Source-channel model for machine translation, Wei [107] claimed that this fails to identify some important transcription probabilities. They then introduced a Direct-Model translation approach that has an increased performance of 4% over Virga and Khudanpur [108].

As identified in the previous section, machine translation systems require a large amount of training data to perform optimally. Since web contains enormous dataset of languages, Jiang et al. [109] had proposed an approach to improve the transliteration model with web data. They employed a Maximum Entropy model that ranks the translation candidates by combining pronunciation similarity and bilingual contextual occurrence in data scrapped from the web.

It should be noted that the above-mentioned approaches are centered on English to Chinese and other European languages. There has been no research conducted

so far with regards to Tamil-Sinhala NE translation. However, Ekbal et al. [110] had proposed a pattern-based approach for named entity transliteration for Bengali to English. They divided both the English and Bengali named entities into transliteration units. Then the system learns the mapping automatically from the bilingual training set to generate co-locational statistics. Their Bengali to English transliterations exhibited a Word Agreement Ratio (WAR) for 81.4%.

While the above-mentioned approaches are based on SMT, Sennrich et al. [111] proposed an approach based on NMT by encoding unknown words as sequences of subword units. It is based on the intuition that various words including names are translatable via smaller units. They had attempted word segmentation using n-gram modes and *byte pair encoding*. They empirically proved that their approach improves the BLEU score over a back-off dictionary-based line by 1.1 and 1.3 for English-to-German and German-to-English respectively.

2.5.6 Summary

Though NMT has been state-of-the-art approach in current times, as shown by Tennage et al. [23]'s research, they require large amounts of parallel data to perform optimally. For the same dataset, Farhath et al. [24]'s SMT approach yielded 20 BLEU score points over NMT. Hence, we can conclude that with current resource limitations surrounding Tamil and Sinhala, SMT is the ideal candidate to perform MT. There has also been no research with regard to integrating NEs in for Tamil-Sinhala MT.

Chapter 3

Methodology

As discussed in the previous chapter, related literature suggests that named entity translation helps improve the overall quality of the translation [107, 108, 110]. But in order to achieve that, we first need to identify the NEs. While there have been NER approaches targeting general domain [13, 14, 16], we are particularly interested in official government documents.

Official government documents contain different types of entities that are usually not present in the general domain. Designations and Government organizations are some examples. The usual general domain tag set consists of Location, Person and Organization. Since we were interested in extracting named entities native to official government documents, which meant that we had to come up with our own tag set.

Once the tag set was identified, the next step was to build an annotated corpus. As illustrated in the literature review, only unsupervised approaches [46, 48] do not require annotation. However, it is hard to define a custom tag set and the performance also has been underwhelming.

Deep-learning based approaches [12, 52] are language and domain independent. They have also shown great performance output making it an ideal candidate for building the named entity recognizer. However, it comes with a caveat; it requires a large corpus.

Building a large corpus manually is an arduous task that consumes time and requires a lot of human effort. On the other hand, if we could annotate a small set, train a machine-learning model, annotate a large corpus using the model, and then employ human annotators to clean the annotated data, it would be relatively easy to achieve. We needed a machine learning approach that would work well with a low amount of labeled data. As discussed in the literature, the ideal candidate to achieve that would be Graph-based Semi Supervised Learning (GSSL).

The challenge of GSSL was to build a graph that captured the local context of words effectively. In order to achieve that initially we represented each occurrence of words with its distributional semantic vector. Though it gave promising results, it fails to capture the context. For instance consider the example "Ministry of Education" and "Secretary of Education". In the first case "of" is part of an *Organization* entity while for the latter its part of a *Designation* Entity. So, we used a novel approach of concatenating n-gram vectors and employed a classification algorithm on the graph.

Once the corpus was large enough, we employed the deep learning approach to finally build the NER model for Tamil and Sinhala. Figure 3.1 illustrates the whole process in a flowchart.

In order to increase the translation performance, we devised an approach to extract the morphological rules in both the languages and identify the root words. We built a translation module that encompassed the rules and the NER model to increase the performance of the existing SMT system.

This chapter details each of the aforementioned steps in detail. Section 3.1 introduces the tag set used. Section 3.2 details the process of building an annotated corpus using the data. Section 3.3 shows how the GSSL approach was used to create the dataset and eventually build a BiLSTM CRF model. Finally, Section

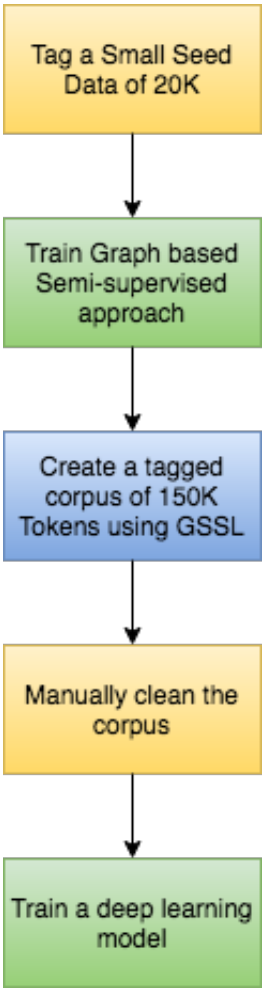

FIGURE 3.1: Outline to build the NER

3.4 explains the steps taken towards integrating NER translation to the existing SMT system.

3.1 Identifying the Tag Set

Major NER research has centered on using a limited number of named entity types like *Person*, *Organization* and *Location*. As revealed in the literature, much of the research for Sinhala and Tamil NER has centered on a general domain with varying tag set. But with different domains, the tag sets also vary.

Since our research centered on official government documents, our initial challenge was to identify a domain-specific tag set. Unlike general domain, in this domain we encountered different types of repeating entities like Designations and Government Organizations. Although a fine-grained approach would have yielded a more detailed information, it would have required a lot of annotated data and expert help.

Hence we decided to choose a coarse-grained approach that would align with the resources at hand. Examining the existing SMT based system revealed that it was struggling to translate names, designation and organization names. We also had a bi-lingual list of government organization and designations names such as: *Ministry of Education*, *Pradhesiya Sabha* and *Secretary*. Hence we devised the tag set shown in Table 3.1.

Tag	Examples
PERSON	**சிவானி** - සිවානි **சிவயோகநாயகி இராமநாதன்** - සිවයෝගනායගි රාමනාදන්
DESIGNATION	**நிதி உதவியாளர்** - මූල්ය සහකාර **விடுமுறை முகாமைத்துவ உதவியாளர்** - නිවාඩු කළමනාකරණ සහකාර
ORGANIZATION	**பிரதேசச் செயலகம்** - ප්රාදේශිය ලේකම් කාර්යාලය **சுகாதார சேவைகள் திணைக்களம்** - සෞඛ්ය සේවා දෙපාර්තමේන්තුව
LOCATION	**யாழ்ப்பாணம்** - යාපනය **கொழும்பு** - කොළඹ
TEMPORAL	18/02/1991 **ஜூலை** - ජූලි
OTHER	**ஐ.நா.** - UN

Table 3.1: NER Tag set

3.2 Annotated Dataset

The Tamil and Sinhala NER corpus currently has over 290K and 210K tokens respectively. We employed 3 human annotators each for Tamil and Sinhala for corpus cleaning purpose. Table 3.2 shows the Kappa statistics for each language.

Language	Kappa Score
Tamil	91.28
Sinhala	89.76

TABLE 3.2: Corpus Kappa scores

The high Kappa scores indicate that there was significant agreement within taggers. One disagreement that presented multiple times was when locations came in combination with organizations. Consider the example, **இலங்கை பாராளுமன்றம்** (Sri Lanka Parliament), some taggers considered the whole thing as an *Organization* while some broke it into Sri Lanka as *Location* and Parliament as *Organization*.

31.1% and 29.9% of total tokens were tagged as named entities for Sinhala and Tamil respectively. Figures 3.2 and 3.3 illustrates the named entity tag distribution in each corpus. As evident from the charts, both corpus have a similar tag distribution.

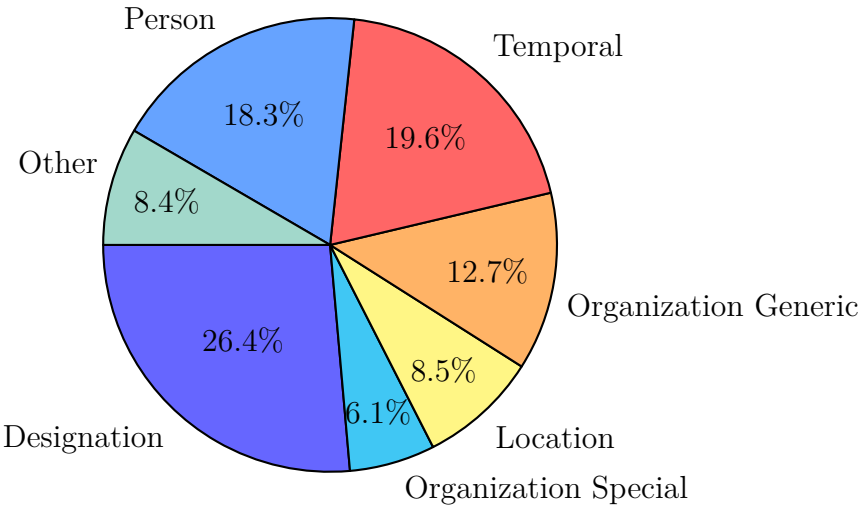

FIGURE 3.2: Named entity tag distribution for Sinhala

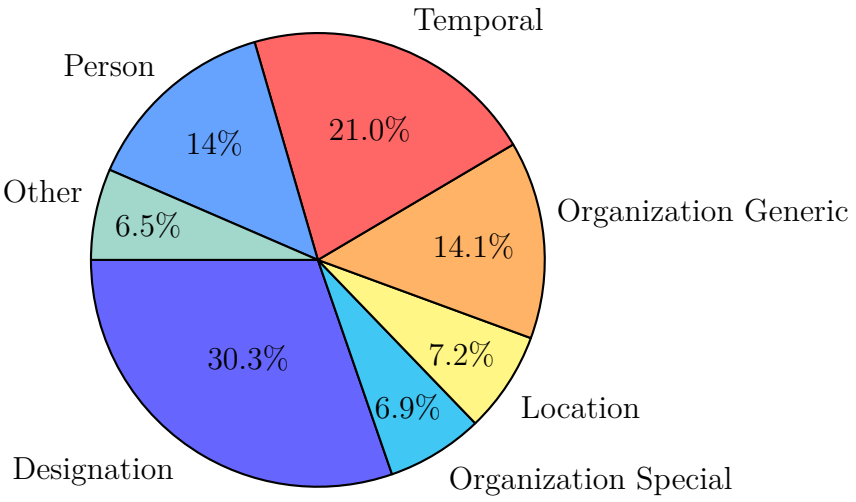

FIGURE 3.3: Named entity tag distribution for Tamil

3.3 Building the Named Entity Recognizer for Tamil and Sinhala

3.3.1 Graph Based Semi-Supervised Learning

3.3.1.1 Representing Nodes of Graph

In sequence tagging problems, the label of a word is predominantly determined by its context. Thus, syntactic relationships between word tokens play a major role. For example, the word *present* may appear as a noun or a verb, depending on the context. Thus, without referring to the context, the exact POS tag of the word cannot be determined. As an example, with respect to Named Entities (NEs), consider the NEs "Central Bank spokesman" and "The Central African Republic". Here, the word 'Central' is used as part of both an Organization and Location [12].

As opposed to using lexical units proposed by Mihalcea and Radev [70] or simple word vector representations proposed by Subramanya et al. [40] to create nodes which are incapable of capturing the local context information paramount for our

sequential tagging. We experimented with different types of vector representations.

The related literature present contradicting arguments with respect to the performance of count and predict models. Baroni et al. [78] and Mikolov et al. [112] claim that predict models such as Word2Vec and FastText capture more syntactic and semantic information compared to traditional count-based distributional models such as PMI vectors. However, Levy et al. [81] have claimed that with proper system choices and hyper parameters, traditional count models can yield similar gains. Recently, Mikolov et al. [87] also claimed that FastText yields a higher gain than GloVe. It should be noted that in count models, increasing the unlabeled data produces extremely sparse vectors that lead to computationally demanding graph building. Thus, we experimented with FastText, Wang2Vec and ELMo. The latter two approaches claim to effectively capture context information that is why we chose to use it with our experiments.

As mentioned earlier, we base our work on one assumption that words with the same local sequence context will have the same sequence tags. In order to capture the local context information in our graph, we experimented with one solution: concatenation of vector n-grams.

3.3.1.2 Creating Edges of the Graph

Similar to the approach proposed by Subramanya et al. [40], once the nodes in the graph are fixed, the edge weights w_{ij} between them between two vector n-grams i and j are defined as shown in Equation 3.1.

$$w_{ij} = \begin{cases} sim(i,j), & \text{if } i \in K(j) \text{ or } j \in K(i). \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Here, $K(i)$ in the set of k-nearest neighbors of vector n-gram i. The similarity function was defined using the Gaussian kernel denoted in Equation 3.2 [113]. Here $d(x_i, x_j)$ is the Euclidean distance between vectors i and j.

$$sim(i, j) = exp(\frac{-d(x_i, x_j)}{2\sigma^2}) \tag{3.2}$$

Theoretically, there can be an edge between each pair of nodes in the graph. However, one can safely disregard edges that have very low weights, because the relationship between such nodes is very weak. Such weak edges can add noise to label propagation.

The identification of the set of vertices that should be connected to a given vertex can be modelled in the form of the k-nearest neighbor problem, where the objective is to determine the set of vertices that have the strongest relationship with the given node (i.e., we determine the set of edges with the highest weight for a given node). Determining the set of edges using k-nn is more effective if the vertices belonging to different classes are well separated. Thus, we transform the vector space to increase the separation of classes.

This dimensionality reduction serves another purpose. The performance of nearest neighbor algorithms degrades when the size of the vector increases. Since we used word-embedding models, it results in 300 dimensions. When concatenating vector n-grams, this dimension reaches 900. Thus, the dimensionality reduction makes graph construction extremely efficient.

Algorithm 1 presents the graph construction procedure.

Algorithm 1: GSSL using word embedding

Data: Corpus with n number of words where n_l are labeled ($n >>> n_l$)

for *each* w_i *in corpus* **do**

$vec_i = ConvertWordToVector(w_i)$;

$v_i = Concatenate(vec_{i-1}, vec_i, vec_{i+1})$;

end

$V_r = BuildVectorList(v)$;

$V_s = SupervisedReduction(V_r)$;

for *each* v_i *in* V **do**

$e_i = NearestKVectors(v_i, distance =' euclidean')$;

$w_i = CalculateWeight(e_i)$

end

$E = BuildEdgeMatrix(e)$;

$W = BuildWeightMatrix(w)$;

Build graph $G = (V, E, W)$;

$Predict(G, n)$

3.3.1.3 Label Propagation

Label propagation refers to the process of assigning labels to unlabeled nodes using the labeled nodes. The prior assumption of semi-supervised learning is that nearby points and points on the same structure are likely to have the same labels [114]. This is a simple and straightforward approach that has been the staple of semi-supervised learning and has yielded encouraging results.

3.3.2 Bi-directional LSTM CRF Sequential Tagging

Introduced by Huang et al. [52], Bi-directional LSTM CRF sequence tagging approach is the baseline for the current state-of-the-art sequential classification.

The current best approach proposed by Peters et al. [12] uses BiLSTM CRF in tandem with their ELMo model to achieve the highest performance for NER and Chunking in English.

However, as noted in the literature review, deep-learning approaches such as this requires a high amount of labeled data. We did not have enough data at the inception of our research. But now with GSSL, we were able to efficiently generate new labeled data. Hence, we chose to implement this approach to build our NER model.

In a sequence-tagging task, we need to access both the past and future inputs at a given time. In order to achieve that, we can utilize a bidirectional LSTM network illustrated in Figure 3.4 as proposed by Graves et al. [115]. Using this approach, we can effectively capture the past and future features.

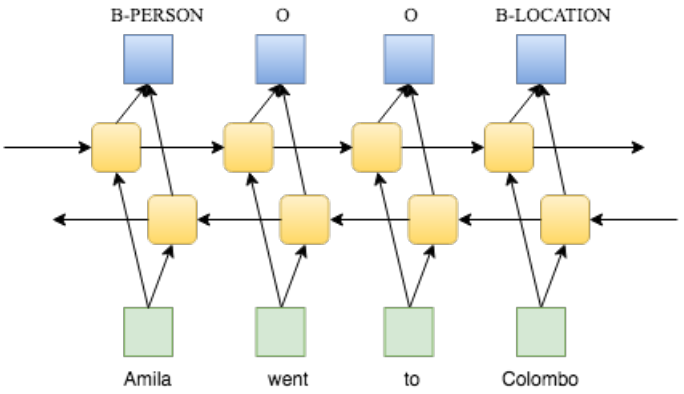

FIGURE 3.4: A bidirectional LSTM network

In order to make use of the neighbor tag information in predicting current tags, we combine the LSTM network with a CRF network to form a LSTM-CRF model, as shown in Figure 3.5. Here, the CRF layer is represented by lines connecting to consecutive tags. This model can efficiently capture the past and future input features using the LSTM layer and sentence level tag information using the CRF layer.

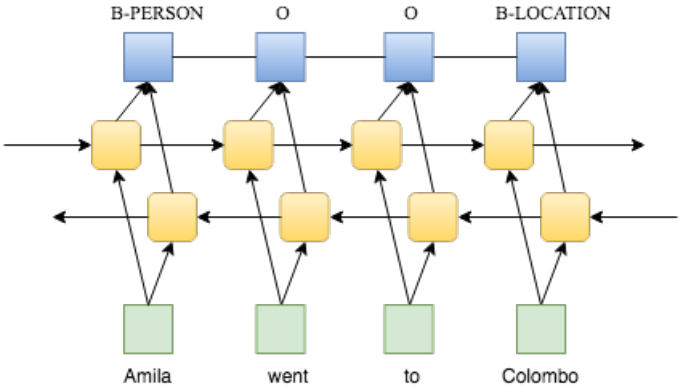

FIGURE 3.5: A BiLSTM-CRF model

Based on this model, we built our approach as illustrated in Figure 3.6. As shown in the diagram, each word in a sentence is mapped to a Word Embedder *(Word Emb)* and Character Embedder *(Char Emb)*. The word embedder uses pretrained word embedding models to convert word to vectors. Similar to GSSL, due its efficiency in capturing semantic and semantic information, we used Wang2Vec [84], FastText [86] and ELMo [59] for its purpose.

On the other hand, the Character Embedder captures character information. Each character in a word is mapped to a randomly initialized embedding. There have been different approaches [53, 54] proposed to capture this information. The following section details these approaches and how we overcame the challenges while adapting it to Sinhala and Tamil.

3.3.2.1 Character Embedding

Convolutional Neural networks: Ma and Hovy [54] had proposed to use CNN to capture the character information in words. They have claimed that CNNs better capture the morphological information like prefixes or suffixes of a word. For this approach, as shown in Figure 3.7 [55], a convolution with 30 filters and filter length of 3 was used, which was then fed to a max-over-time pooling.

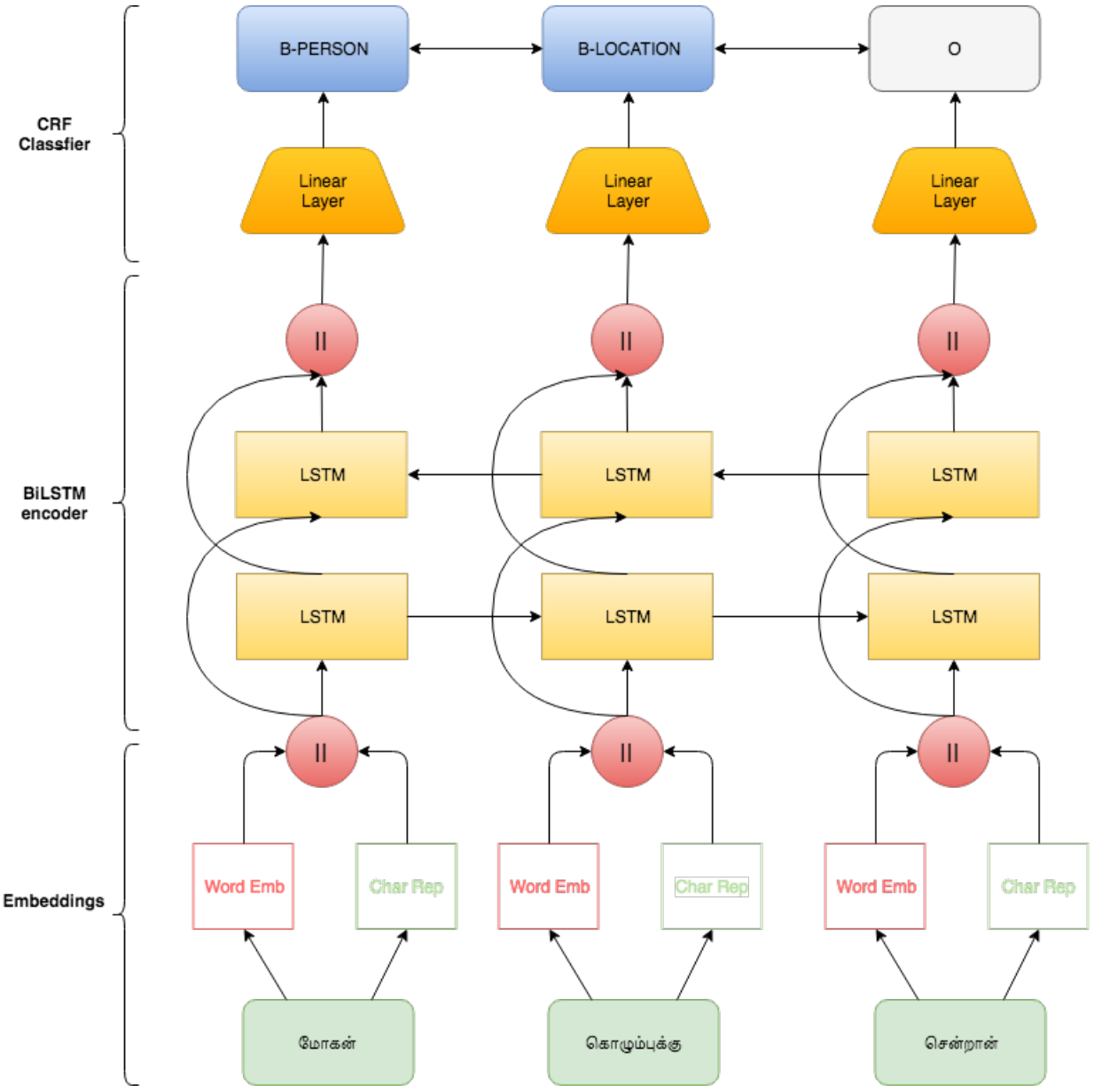

FIGURE 3.6: Architecture of the BiLSTM network with a CRF Classifier

BiLSTM Networks: Lample et al. [53] had proposed a BiLSTM-based encoder to create character embedding. For this approach, as shown in Figure 3.8 [55], the character embedding is fed into a BiLSTM encoder and the last output of the two LSTM networks are then combined to form a character-based representation.

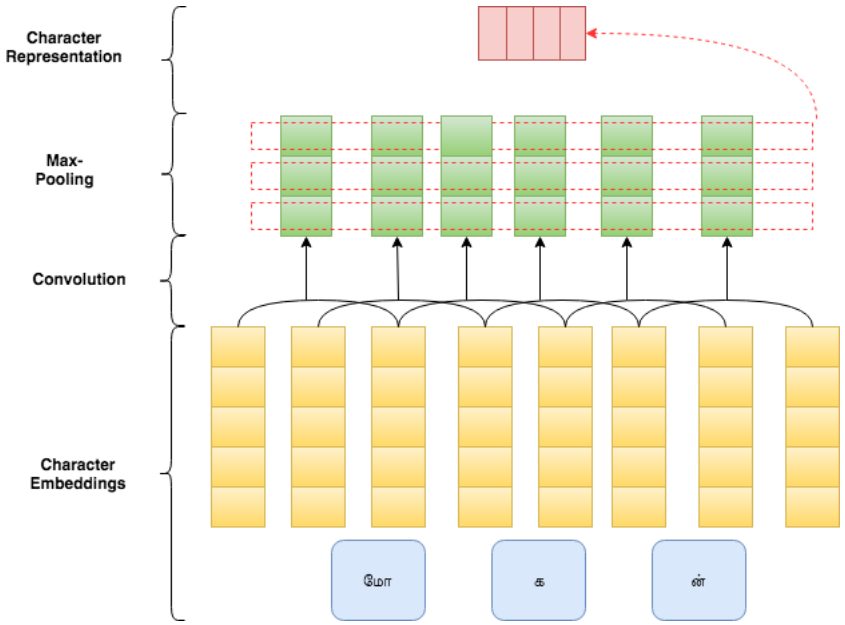

FIGURE 3.7: Character-based representation using convolutional neural network

One of the key challenges with character-based representation with Tamil or Sinhala is the Unicode-processing problem. For example, consider the word மோகன், when broken into individual characters it will result in ம, ோ , க, ன, ். This will result in erroneous character based information extraction. Instead we need to extract individual characters like மோ ,க ,ன் . To achieve this, we used an existing utility package for Tamil [116]. For Sinhala we created a utility function that would break each word into its Sinhala character units. For example it would break the අම්ලට to අ ,ම් ,ල ,ට.

3.3.2.2 Predicting the Tags

Once the character-based representation is extracted, as shown in Figure 3.6 [55], we concatenate the word embedding and the character vector to be used for the BiLSTM encoder. One LSTM network runs from the beginning to the end of the sentence and the other in reverse. The outputs of both forward and backward LSTM networks are concatenated and used as input for the Softmax classifier. This classifier gives the probability distribution of the tags. Then, the tag with

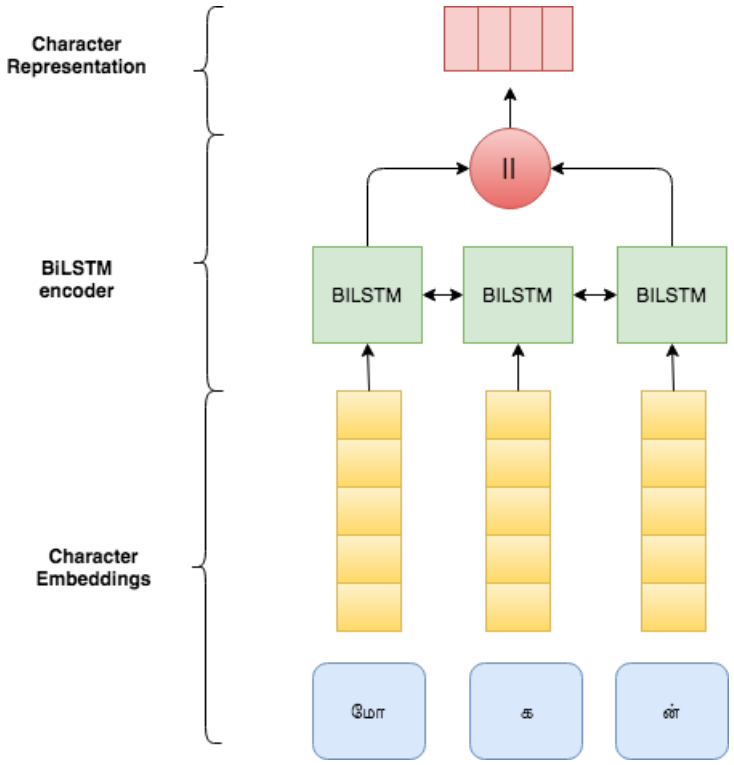

FIGURE 3.8: Character-based representation using BiLSTM networks

the highest probability is selected where the linear-chain CRF maximizes the tag probability of the sentence [55].

3.3.2.3 Tuning the Hyper-parameters

The BiLSTM CRF approach comprises of many hyper parameters including the pre-trained vectors and character representation that yield different results. Following are some parameters:

Optimizer Optimizer is used for the minimization of the objective function [55]. A commonly used one is Stochastic Gradient Descent (SGD). Though SGD has been used in a large number of machine learning systems, it has been sensitive towards the selection of learning rate. Hence, as an alternative Adagrad [117], Adadelta [118], Adam [119] and Nadam [120] were used.

Gradient Clipping and Normalization Vanishing and exploding gradient problems are two widely-known problem with RNN [121]. LSTM networks counter the vanishing problem, while gradient clipping [122] and gradient normalization [123] had been proposed to solve the exploding problem. We evaluated both of them.

Number of Recurrent Units Number of recurrent units used within each LSTM layer. Typical choices are 25, 50, 75 and 100 [55].

Number of LSTM-Layers Number of LSTM layers used within forward and backward layers. Typically experimented up to 3 [55].

Batch Size Training batch size per iteration. Typical choices are 1, 8, 16, 32 and 64 sentences [55].

3.4 Translating Identified Named Entities

Once the named entities have been identified, the next task was to translate them and improve the flow of the text for a better translation. Both Tamil and Sinhala are highly inflectional languages. For example a name අමිල / அமில can occur as අමිලට /அமிலவிடம் or අමිලගෙන්/அமிலவால் depending on the context. The first challenge was to identify the morphological transformations and extract the root word. Section 3.4.1 details a method that identifies morphological transformations in an unsupervised manner. This approach extracts the morphological rules for named entities.

Section 3.4.2 details how we integrated named entity recognizer and extracted morphological rules to the the Moses system to increase the translation performance.

3.4.1 Unsupervised Morphology Induction

Since both Tamil and Sinhala are morphologically rich, one of the key challenges with translation was to identify the inflection of a word and extract the root form. Soricut and Och [124] had proposed an unsupervised morphological induction approach that extracts morphological candidates from a big corpus and generates lexicalized morphological information. They analyzed every word from a corpus to identify all possible morphological transformations. Their approach did not account for root names. Unlike them, we already had a list of root names extracted from the government of Sri Lanka's database. While Soricut and Och [124] identified the morphological transformations by building a graph, we implemented a simple threshold approach.

Lets say the list of names is N and the list of vocabulary from extracted Wikipedia and official government documents being is V. We followed the following steps.

1. Extract candidate-specific morphological suffix transformations from N to V
2. Identify common rules from the candidate set
3. Extract all possible translation mappings for the morphological transformations between Sinhala and Tamil

Starting from w_1 in N and w_2 in V, we used the algorithm proposed by Soricut and Och [124] in combination with the Sinhala and Tamil utility function to extract all possible suffix substitutions from w_1 to w_2. We decided to only extract suffix substitutions and not focus on prefix because there were very rare cases of prefix substitutions for Tamil and Sinhala.

We denoted those substitutions as *from:to*. For instance ϵ:இடம் / ϵ:ට the addition of இடம்/ ට ; this can been seen in examples like மோகனிடம் / අමිලට, செயலாளரிடம் / නිලධාරියාට. Here the ϵ denotes no alterations from root word.

We then used the frequency score of every rule and clustered them into two sets. The rationale behind this approach is that high frequency ones are inducted as the rules. Finally, we extracted all possible name translations along with their rule translations which will be incorporated to the SMT system.

3.4.2 Integrating to Moses

Once we extracted the named entities and identified the morphological transformations, the next step was to integrate them to the existing Moses system. Moses provides an option to use external data for translation. For example, with number and named entities, we can plug translations to the decoder without changing the model.

We based our approach on the hypothesis that replacing the words that do not exist in the parallel corpus, i.e: *Unknown words*, with similar meaning words will increase the accuracy. Koehn et al. [25] had shown that replacing unknown words help increase the flow of translation.

We already had a list of translated named entities obtained. First we identified named entities that does not exist in the parallel corpus and extract the root words. If the root word exists in parallel corpus we replace the original word. If the root word does not exist in parallel corpus but is in the list of translated named entities, we will externally specify the translation. Otherwise, we will replace the original word with the nearest word vector and specify the transliteration as its translation. The flowchart in Figure 3.9 details our preprocessing steps.

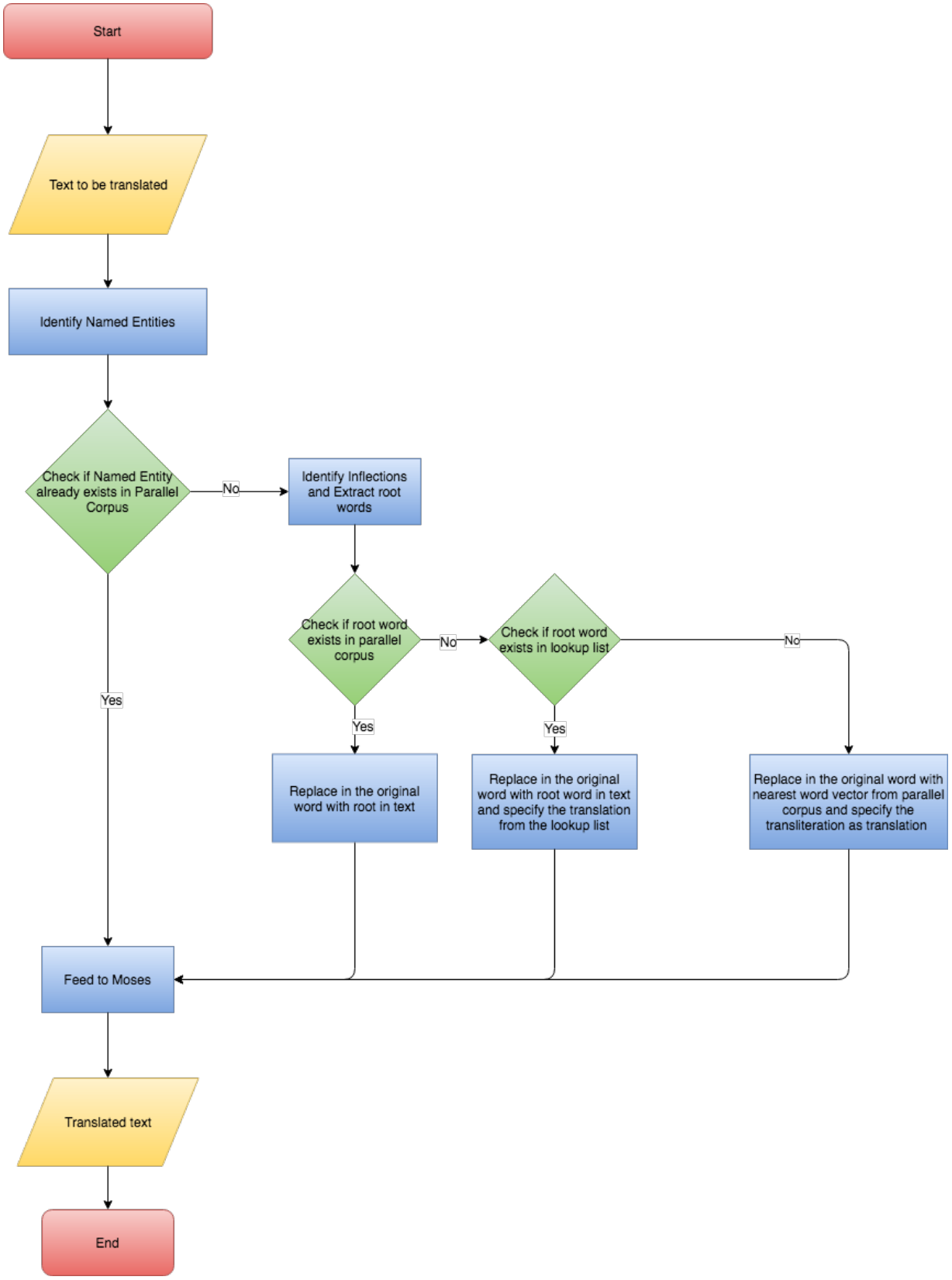

FIGURE 3.9: Preprocessing the input data

Chapter 4

Implementation

This chapter briefly explains the key implementation details. Section 4.1 details the corpus building process carried out and the tool that was used. Section 4.2 details the process carried out with regarding to build the GSSL and BiLSTM CRF approach for NER. And finally 4.3 briefly details the integrationto Moses and introduces the Si-Ta translation system designed by the University of Moratuwa. One of the key goals of our research is to increase the performance of the existing system.

4.1 Building the Corpus

One of the initial challenges of our research was to build a named entity corpus. We used the Tagtog [125] annotation tool for this purpose. Figure 4.1 illustrates a typical annotation window.

4.2 Building the Named Entity Recognizer

4.2.1 AllenNLP Research Library

AllenNLP [126] is a powerful open-source research library built on top of PyTorch. It makes it easy to design and evaluate deep learning models for any NLP problem.

FIGURE 4.1: The Tagtog annotation tool

Its pre built support for CUDA GPU systems makes it effective and faster to build models. For our purpose of building Named Entity and deep contextual vectors, we modified the code and added new models. The code is currently being housed in the UOM-Allen [127] repository.

4.2.2 Building the Word embedding Models

ELMo As powerful as it is, the ELMo model requires a lot of time and processing power to train. The code [128] was already available. We created a Google cloud instance with 4 Tesla V80 GPUs, 16 CPUs and 40GB memory to train the model. We ran the training fro 3 Epochs. It took over 4 days to train 27M tokens and over 8 days to train 60M tokens for Tamil and Sinhala respectively. The dimension of vector produced was 1024.

In order to compare our GSSL approach with the existing state-of-the-art approach for low-resource settings, we also obtained the English language model as well. It should be noted that the English language model was a pre-trained model obtained from the Github AllenNLP repository [126].

A good language model is one that best predicts an unseen test set. Perplexity score is the inverse probability of the test set, normalized by the number of words [129]. Thus, lower the score the better the model. Table 4.1 lists the perplexity score obtained for each language. It is evident that Sinhala and Tamil scores are higher than English. This is because of the smaller amount of training data available than English and running the training model only for 3 epochs opposed to 10. We were able to run only 3 epochs because of the lack of processing power available.

Language	Perplexity score
English	27.67
Sinhala	87.67
Tamil	112.27

Table 4.1: Perplexity scores for ELMo Model

FastText We used pretrained models [130] provided by *facebookresearch* for our training purpose. These vectors of 300 dimensions were trained using the Wikipedia data dump.

Wang2Vec The code [131] to create vector modes was freely available. We experimented with the *CWindow* and *Structured Prediction* approaches provided. Though there are no discerning advantages over each other, structured prediction was giving a very slight performance benefit. Unlike ELMo, there weren't any particular processing equipment required to create this model.

4.2.3 Modifying the metric-learn library

metric-learn[132] is an open-source library that provided the dimensionality reduction approaches used for our research. Our research was concerned with distance measurement for large amounts of data. The existing code was not optimized to handle such cases. This was due to the use of *euclidean_distances* function from *sklearn* in the original code.

As shown in the following code snippet, it is the slowest one with 6.78s. Though *scipy* and *numpy* libraries are faster, the fastest one is using the native math library provided by Python. It runs in 0.42s compared to 0.65s of *numpy* and 1.53s of *scipy*. We modified the original code of metric-learn to reflect this finding and optimized the code.

```
1 import numpy as np
2 from scipy.spatial import distance
3 from sklearn.metrics.pairwise import euclidean_distances
4 import math
5
6 np.linalg.norm(v1-v2) #0.654838958307s
7 distance.euclidean(v1, v2)#1.53977598714s
8 euclidean_distances(v1, v2)#6.7898791732s
9
```

```
dist = [(a - b)**2 for a, b in zip(v1, v2)]
dist = math.sqrt(sum(dist))
return dist #0.422228400305s
```

LISTING 4.1: Distance measurement efficiency

4.2.4 Implementing Graph Based Semi-supervised Sequential Tagging Algorithm

One of the key challenges with implementing graph based semi-supervised approach was the high dimensionality and large number of vectors. High dimensionality of the vectors and the large size of the sample space severely affected the performance of the k-nn algorithm. Thus, we resorted to approximate nearest neighbor algorithms (ANN) which performs faster than traditional nearest neighbor algorithms for large data set size. We use Annoy [133], which has been empirically shown to work better with large data sets [134]. k was set to an arbitrary value of 20. It should be noted this ANN's accuracy drops when dimensions of the vector are greater than 100. This attribute played an important role in choosing to reduce dimensions.

To achieve a discriminant feature set in a lower dimension, two dimensionality reduction techniques were experimented with: Linear Discriminant Analysis (LDA) and Local Fisher Discriminant Analysis (LFDA). Both LDA and LFDA are supervised methods that are useful in finding dimensions that aim at separating the clusters [135].

For label propagation, Harmonic Function (HMN) [114] and Local and Global Consistency (LGC) [136] were experimented with. These are two well-established label propagation algorithms that have proven their effectiveness in different contexts [71]

4.2.5 Implementing BiLSTM CRF Tagging

We ran the *CRF Tagger* provided in the AllenNLP inside a Docker to build the tagger. To run the model, we created a Google cloud instance with 2 Tesla V80 GPUs, 8 CPUs and 20 GB of memory. *Appendix B* lists the configuration file used.

4.3 Integrating to Moses

The decoder has an XML markup scheme as shown in the following listing allows to specify translations for parts of a sentence. In its simplest form, we can tell the decoder what to use to translate certain words or phrases in the sentence [137]. Here we tell the module not to translate the phrase *ein kleines haus* or *haus* and instead use the one specified externally as *a cute place* and *dwelling* respectively.

```
1 echo 'das ist <np translation="a cute place">ein kleines haus</np>'
    \
2   | moses -xml-input exclusive -f moses.ini
3 this is a cute place
4  echo 'das ist ein kleines <n translation="dwelling">haus</n>' \
5   | moses -xml-input exclusive -f moses.ini
6 this is a little dwelling
```

Listing 4.2: Defining external translation for Moses

4.3.1 SiTa System

The translation system we are planning to integrate our work is the *Si-Ta* system. *Si-Ta* is a machine translation system developed by the University of Moratuwa. The goal of the system is to aid translators with translating official government documents associated with government entities. Funded by the Department of the Official Language System, the project has been ongoing since January 2015.

The system is built using the Moses translation system and human translators have reported a score of 3.32 on a scale of 1-5.

Chapter 5

Experiments and Results

This chapter details the experiments, results and analysis with regards to the research carried out. This section has been divided into 3 main sections.

Section 5.1 details the experiments carried out with the proposed GSSL approach. The section reports on the effectiveness of local context captured by vector n-gram concatenation.

Section 5.2 reports on the BiLSTM CRF approach. Since this approach is language independent, we implemented our approach for not only for NER but also for POS tagging as well.

Finally, section 5.3 details the experiments carried out on integrating the NER to the Moses system. It details how each of our hypotheses affect the performance of the existing system.

5.1 Graph Based Semi Supervised Learning

Dataset

English. We evaluated our approach on CoNLL2003 NER task [1] for POS, NER and Chunking task. We emulated a low-resource setting for English by using only 20K, 40K, 60K and 100K data as our training setting as opposed to using the full training data. We carried out testing for English to benchmark our results against the state-of-the-art approach for English.

Tamil. For Tamil POS tagging we used the dataset from the Forum for Information Retrieval (FIRE) [29] and for NER we used the official government corpus we had built. The POS dataset has nearly 80K labeled data with 32 POS classes.

Sinhala. For our experiments, we used the University of Moratuwa (UOM) Sinhala POS corpus [138], which currently has 260K tagged tokens labeled using 32 tags. For NER we used the official government corpus we had built.

Experiment Setup

Experiments are designed to determine the impact of local context information in graph construction for sequential tagging tasks, and the impact of dimensionality reduction on the same. For English, we test the performance of our solution with respect to POS tagging, NER, and Chunking tasks of the CoNLL 2003 dataset. With respect to Tamil and Sinhala, we experiment with POS tagging and NER.

The current implementation employs the Continuous Bag of Words (skip-gram) model of FastText [57] to generate word embedding for English, where the vector dimension is 300. Wang2Vec models are generated using a part of the wiki dump for all three languages. Dimension of these vectors is also set to 300. The ElMo model [12] of 1024 dimensions was used.

We have experimented with $n = 3$, when generating vector n-grams. For example, when $n = 3$, the word "Central" will be represented by concatenating the

word vectors of “The”, “Central”, “African”, thus adding the context information. Thus, we end up with a feature vector of 900 dimensions for FastText and Wang2Vec, and 3072 for ElMo.

For each language, the graph is constructed using 1 million tokens from an unlabeled corpus, and the labeled text size is varied from 20k to 100k in a step-wise manner.

To show that our GSSL solution works in low-resourced settings better than the state-of-the-art reported in the context of high-resourced settings, we compare our results with the work of Peters et al. [12]. We sampled the same amount of training samples from the CoNLL corpus.

For this experiment, according to the discussion by Peters et al. [12], we used two bidirectional GRUs with 80 hidden units and 25 dimensional character embedding for the token character encoder. The sequence layer uses two bidirectional GRUs with 300 hidden units each. For regularization, we add 25% dropout to the input of each GRU, but not to the recurrent connections to setup the model. We also embed the ElMo model to represent each word in this bidirectional model and tested it.

Results

For POS, we report the accuracy, while for Chunking and NER, we report the official evaluation metric (micro-averaged F1 score).

	POS					Chunking					NER				
	20K	40K	60K	80K	100K	20K	40K	60K	80K	100K	20K	40K	60K	80K	100K
FastText															
A	0.75	0.79	0.83	0.839	0.81	0.66	0.70	0.73	0.73	0.71	0.35	0.30	0.46	0.46	0.34
B	0.69	0.72	0.74	0.77	0.74	0.66	0.69	0.67	0.72	0.74	0.31	0.25	0.44	0.43	0.35
C	0.60	0.64	0.68	0.70	0.66	0.53	0.57	0.57	0.69	0.60	0.38	0.30	0.44	0.46	0.39
D	0.85	0.88	0.87	0.88	0.86	0.79	0.83	0.85	0.83	0.83	0.61	0.53	0.69	0.66	0.50
ElMo															
A	0.84	0.84	0.88	0.88	0.86	0.82	0.85	0.85	0.82	0.84	0.70	0.67	0.84	0.81	0.65
B	0.90	0.91	0.92	0.92	0.91	0.82	0.83	0.84	0.83	0.84	0.69	0.65	0.76	0.79	0.70
C	0.74	0.76	0.83	0.81	0.77	0.76	0.80	0.79	0.78	0.78	0.62	0.56	0.81	0.77	0.57
D	**0.928**	**0.934**	**0.941**	**0.942**	**0.93**	**0.90**	**0.91**	**0.92**	**0.88**	**0.90**	**0.79**	**0.76**	**0.86**	**0.89**	**0.70**

TABLE 5.1: Comparison of different methods to represent nodes and their respective accuracy for different tasks in English. A - Single Vector, B - Dimension reduced Single Vector, C - Concatenated n-gram vectors, D - Dimension reduced concatenated n-gram vectors.

Both LDA and LFDA showed near-equal performance, and so did HMN and LGC. Thus, the results only showcase the experiment setups that used LDA and HMN.

Table 5.1 shows the impact of different word embedding models in vertex representation, with and without dimensionality reduction on POS, NER, and Chunking tasks in the CoNLL 2003 dataset. It also shows the impact of n-gram concatenation, and dimensionality reduction. Results are reported for different labeled data set sizes, which demonstrate a low-resourced setting.

Since there were no pre-trained embedding available for Wang2Vec, we trained from the first billion characters from Wikipedia for English. This lead to an sub-optimal results across all tasks, hence we have omitted from reporting it.

As indicated by the results in Table 5.1, it is evident that ElMo performs much better than FastText for all the tasks and all the dataset sizes. While n-gram concatenation or dimensionality reduction did not show compelling results when used in isolation, when combined they contributed to a significant performance gain for both FastText and ElMo.

In this experiment, we used Annoy approximate nearest neighbor algorithm to quickly calculate the nearest neighbors. Benchmarks done on ANN [134] have shown accuracy drops when the dimension increases above 100. This can be seen in our results - with concatenated vectors or high-dimension vectors like ElMo the accuracy is considerably lower. Since our approach was transductive, we were wary of the efficiency and timing. Traditional k-NN algorithms may give high scores but will require high memory consumption and days to complete.

Tables 5.2 and 5.3 show the results of similar experiments carried out for Sinhala-Tamil POS and NER tagging tasks.

	Tamil POS			Sinhala POS				
	20K	40K	60K	20K	40K	60K	80K	100K
ELMo								
A	0.76	0.77	0.79	0.85	0.80	0.84	0.85	0.80
B	0.84	0.87	0.87	0.90	0.86	0.89	0.88	0.84
C	0.59	0.62	0.62	0.73	0.65	0.73	0.73	0.74
D	**0.86**	**0.88**	**0.90**	**0.91**	**0.89**	**0.90**	**0.89**	**0.84**
FastText								
A	0.73	0.74	0.75	0.80	0.76	0.83	0.82	0.77
B	0.62	0.79	0.77	0.80	0.77	0.83	0.82	0.79
C	0.54	0.58	0.58	0.66	0.60	0.67	0.66	0.59
D	0.80	0.81	0.83	0.901	0.88	0.88	0.85	0.84
Wang2Vec								
A	0.72	0.74	0.70	0.81	0.77	0.84	0.81	0.77
B	0.59	0.60	0.54	0.78	0.76	0.81	0.79	0.77
C	0.58	0.58	0.57	0.71	0.66	0.70	0.70	0.63
D	0.70	0.71	0.72	0.80	0.76	0.84	0.85	0.81

TABLE 5.2: Comparison of different methods to represent nodes and their respective accuracy for Tamil and Sinhala POS tagging. A - Single Vector, B - Dimension reduced Single Vector, C - Concatenated n-gram vectors, D - Dimension reduced concatenated n-gram vectors.

We then compared the performance of our GSSL approach against Peters et al. [12] using the best result reported in Table 5.3. As shown in Figures 5.1, 5.2 and 5.3, when the ELMo model with n-gram concatenation and dimensionality

	Tamil NER				Sinhala NER			
	20K	40K	60K	80K	20K	40K	60K	80K
ELMo								
A	0.54	0.58	0.61	0.64	0.42	0.54	0.60	0.66
B	0.56	0.61	0.62	0.66	0.46	0.56	0.61	0.67
C	0.37	0.46	0.47	0.58	0.36	0.43	0.45	0.51
D	**0.62**	**0.67**	**0.71**	**0.72**	**0.61**	**0.65**	**0.68**	**0.69**
Wang2Vec								
A	0.26	0.27	0.29	0.31	0.26	0.29	0.29	0.32
B	0.29	0.31	0.32	0.37	0.28	0.30	0.30	0.36
C	0.21	0.23	0.26	0.31	0.20	0.22	0.24	0.30
D	0.39	0.42	0.43	0.45	0.37	0.40	0.43	0.44
FastText								
A	0.30	0.30	0.31	0.34	0.30	0.31	0.31	0.33
B	0.37	0.38	0.40	0.40	0.38	0.37	0.38	0.41
C	0.25	0.25	0.31	0.34	0.22	0.24	0.31	0.31
D	0.44	0.48	0.54	0.54	0.42	0.47	0.53	0.54

TABLE 5.3: Comparison of different methods to represent nodes and their respective F1-scores for Tamil and Sinhala NER tagging. A - Single Vector, B - Dimension reduced Single Vector, C - Concatenated n-gram vectors, D - Dimension reduced concatenated n-gram vectors.

reduction is used, our GSSL approach outperforms Peters et al. [12]'s bidirectional LSTM CRF.

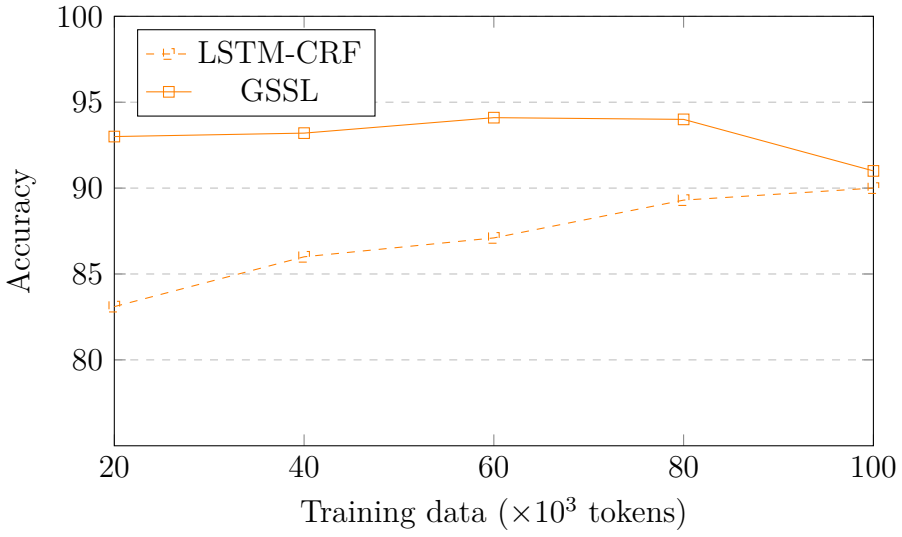

FIGURE 5.1: English POS accuracy for GSSL Vs LSTM-CRF

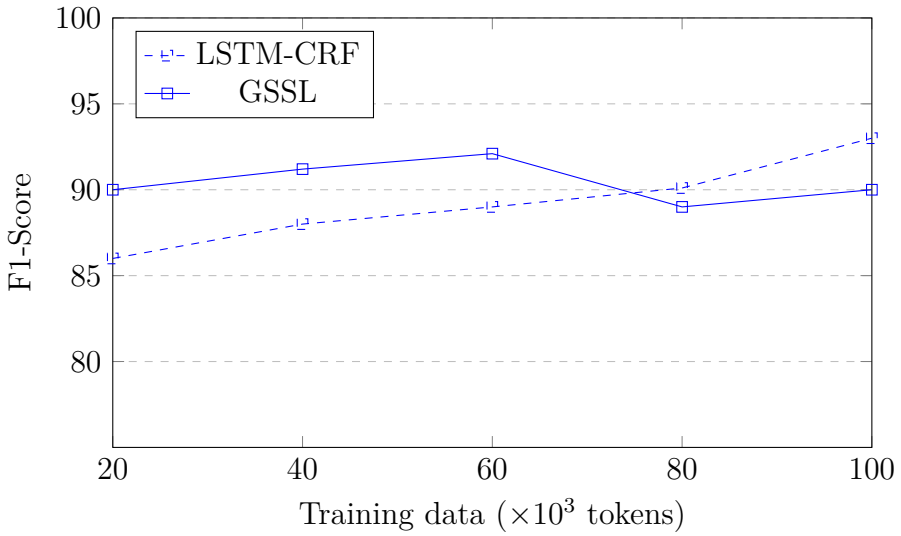

FIGURE 5.2: English chunking F1-Score for GSSL Vs LSTM-CRF

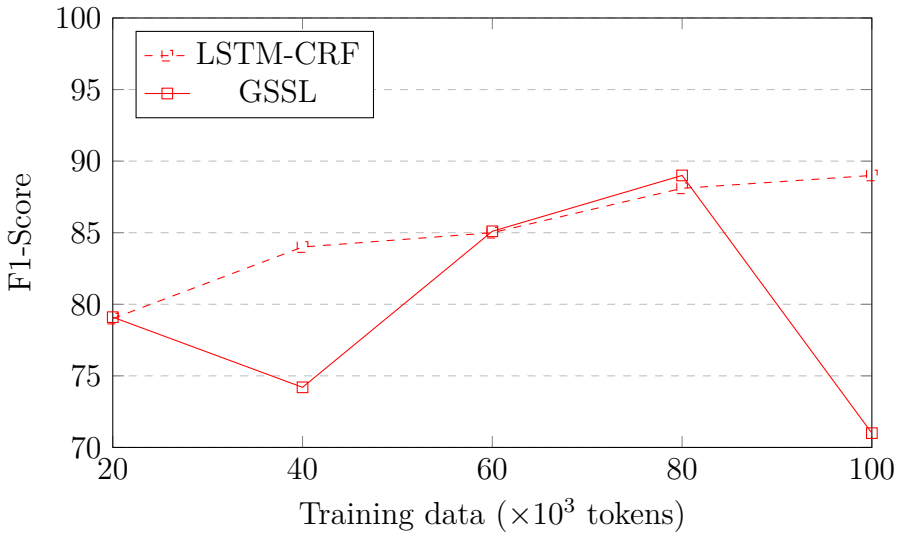

FIGURE 5.3: English NER F1-Score for GSSL Vs LSTM-CRF

Analysis

According to the Figures 5.1 5.2 5.3, when increasing training data, as opposed to our expectations there are some drops in the scores. A glaring one was with NER. For dimension-reduced concatenated ELMo vector with 80K training data resulted an 89.0 F1 score, and it drops to 71.0 for 100K training data. Further

analysis revealed that when the training dataset was increased, it had lead to misclassification due to the nearness of vectors. For example, our training data had *Germany* as *LOC* and the test data had *German*, which was supposed to be classified as *MISC* but was classified as *LOC* due to the close proximity of the vectors.

5.2 Bi-directional LSTM CRF Tagging

5.2.1 NER

The Tamil, Sinhala NER corpus has 290K and 210K tokens respectively. We broke the corpus into 4:1 as training and testing sets. Table 5.4 shows the F1-Score of the NER tagger with different word-embedding models.

Our experiments showed that tuning the hyper parameters such as batch size, dropout value, number of recurrent units and number of LSTM-layers didn't have any discernible advantage. But using Nadam for optimizer with gradient clipping and CNN for character encoding resulted in the optimal output. The data we have report uses these features with 2 LSTM-layers, 100 recurrent units, 64 batch size and 0.25 dropout value.

	F1 Score				
	A	B	C	D	E
Tamil	82.3	85.1	85.2	**85.4**	85.1
Sinhala	79.8	82.8	83.0	**83.1**	83.0

TABLE 5.4: Comparison of different vectors and their respective accuracy for Tamil and Sinhala NER tagging with BiLSTM CRF. A - FastText, B - Wang2Vec, C - ELMo, D - ELMo + Wang2Vec, E - ELMo + FastText

5.2.2 POS

Since the bi-directional LSTM CRF tagger is task and language independent, we also built a POS tagger for Sinhala and Tamil.

Sinhala

The University of Moratuwa already had a POS corpus [138] and tagger. We used the same corpus to build our models and compared against the pre-existing methods proposed by Fernando et al. [138] as the baseline.

We sampled out a 20K dataset form this corpus as training data for both our GSSL approach and Fernando et al. [138]'s approach. The SVM Tagger reported an accuracy of 87.11% while we were able to achieve an accuracy of 91.4%.

For the deep-learning approach, we conducted five different experiments for two different corpus: News and Official Documents (OD). The NEWS and OD corpus consisted of 253,635 and 140,411 tokens respectively. We broke the corpus into 4:1 as training and testing sets. Table 5.5 shows the accuracy of the POS tagger with different word-embedding models. We followed the same experimental setup used in NER.

Training Corpus	Testing Corpus	Accuracy					Baseline
		A	B	C	D	E	
OD	OD	82.1	83.2	85.4	**86.8**	85.8	84.9 (CRF)
NEWS	NEWS	85.2	86.4	88.3	**88.7**	88.4	85.71 (SVM)
OD+NEWS	OD+NEWS	81.4	83.2	87.1	**87.6**	87.1	87.2 (CRF)
OD	NEWS	79.3	79.1	81.0	**82.1**	81.3	75.26 (SVM)
NEWS	OD	82.3	83.1	84.2	**85.1**	84.7	83.9 (CRF)

TABLE 5.5: Comparison of different vectors and their respective accuracy for Sinhala POS tagging with BiLSTM CRF. A - FastText, B - Wang2Vec, C - ELMo, D - ELMo + Wang2Vec, E - ELMo + FastText

Tamil

For Tamil POS tagging, we used the FIRE corpus. It had 80K tokens that were split into 70k for training and 10K for testing. Table 5.6 shows the accuracy of the POS tagger with different word-embedding models. We followed same experimental setup used in NER.

Vector	Accuracy
FastText	91.23
Wang2Vec	94.12
ELMo	94.17
ELMo + Wang2Vec	**94.54**
ELMo + FastText	94.2

TABLE 5.6: Comparison of different vectors and their respective accuracy for Tamil POS tagging with BiLSTM CRF

Analysis

Though deep-learning with distributional semantic vectors yielded better results when compared to other baseline approaches, much to our disappointment, the ELMo vector did not yield the same spike as English. This was due to the lack of monolingual data available for both the languages to train the ELMo models. With limited resources at hand, we were only able to run the training model only for 3 epochs. This was also evident from the high perplexity scores for Tamil and Sinhala. Sinhala had a slightly lower score since it had more amount of data to train from. On the other hand, Wang2Vec [84] produced high results staying true to its claim of being able to capture syntactic information well.

Tamil NER corpus had 290K tokens whereas Sinhala only had 210K tokens. Armed with a higher amount of data and a better Kappa score, Tamil NER resulted in a better score than Sinhala NER.

5.3 Integrating to Moses

Experimental Setup

We used the official government SiTa Parallel Corpus [24] for our research purposes. The corpus consisted of 25,476 sentences, which were divided into 24376 for training, 1000 for tuning and 300 for testing. Giza++ [94] was used for the word alignment with *grow-diag-final-and* as the summarization heuristics. Heafield [139]'s KenLM was used to generate the language model. The tuning of feature weights was done using Minimum Error Rate Training [140]. Finally the testing was done using the Moses decoder and the BLEU score analysis generated using the script provided by Moses.

Table 5.7 shows the different experiments designed as part of the experiment. Here, E3 and E4 are mutually-exclusive experiments

Experiment	Description
E1	Identify NEs that are not in the parallel corpus, if root words exists in list then translate else transliterate
E2	Identify NEs that are not in the parallel corpus but root word exists, replace the identified NEs with root word
E3	Identify NEs and its roots word does not exists in parallel corpus and lookup list, replace the word with nearest word vector
E4	Identify NEs that are not in the parallel corpus but root word does not exists in lookup list, replace the identified NEs with nearest vector word from parallel corpus and specify the original word as translation

Table 5.7: SMT integration experiments

5.3.1 Sinhala → Tamil Translation

Baseline BLEU Score - 35.28

Experiment	BLEU Score
E1	**36.67**
E2	35.78
E3	35.27
E4	36.11
E2 + E4	36.23
E2 + E3	36.60
E1+ E2 + E3	36.60
E1 + E2 + E4	35.67

TABLE 5.8: Sinhala→Tamil translation scores after named entity translation integration

5.3.2 Tamil → Sinhala Translation

Baseline BLEU Score - 23.75

Experiment	BLEU Score
E1	23.90
E2	24.01
E3	24.07
E4	23.80
E2 + E4	24.05
E2 + E3	24.08
E1+ E2 + E3	24.12
E1 + E2 + E4	**24.24**

TABLE 5.9: Tamil→Sinhala translation scores after named entity translation integration

Analysis

Sinhala is phonetically more complex than Tamil and the transliteration module we employed was only a 1-to-1 mapping. Using this approach is reflected with the dipping of the BLEU score while employing the transliteration module for Tamil→Sinhala but increasing for Sinhala→Tamil. Though replacing with root words will not produce semantically ideal translation, it brings more flow to the text and thus increases the score by a relatively large margin. Usage of word embedding and replacing the words with the nearest NE entity from the parallel corpus will yield an incorrect output. Hence, with E3, we replaced the word altogether and specified it as the translation and with E4, we only replaced it to generate a smooth flow while still preserving the original word. As expected, E4 produced a higher score than E3. Though the nearest word vector produced syntactically correct sentences, they were semantically wrong. For example, in Tamil, **பத்திரங்களுக்கு** was replaced with **பாத்திரங்களுக்கு** . This is because both the words have mostly similar sub-word structure. Here the first word means *document* while the latter means *container* thus resulting in semantically wrong translations.

Chapter 6

Conclusion

The aim of this research was to improve the existing machine translation system for Sinhala-Tamil official government documents by incorporating named entity translation. Our initial analysis had revealed the short comings of the machine translation system to translate named entities in Sinhala-Tamil official government documents.

We had proposed a novel tag set specific to the official government documents that captured varying types of recurring named entities. Identifying named entities was challenging given the limited amount of annotated data. In order to tackle it, we proposed a novel graph-based sequential tagging approach that outperformed the existing state-of-the art for low-resource settings. Our solution is based on identifying neural word embedding models that better capture local context information in graph vertices, and producing a graph in a low-dimensional space that has vertices belonging to different classes well separated. While some of the word embedding models employed did not generate the expected result, in general, our hypothesis of capturing context information by concatenating vectors is validated. In particular, n-gram concatenation and dimensionality reduction resulted in significant performance gains. Our novel GSSL solution can be presented as a promising alternative for sequential tagging in low-resource languages.

We were able to effectively build a large corpus using the graph-based approach and manual cleaning. With larger annotated data, we employed Bi-directional LSTM CRF to build the final NER model. This was the first deep-learning approach carried out with regards to Tamil and Sinhala. As part of this approach, we also empirically experimented with different word-embedding models and identified the ELMo approach as the best. Employing the same model, we were also able to build a better POS tagger for Sinhala.

As part of the named entity translation, we were able to build a module that extracted morphological rules in an unsupervised manner. Finally we used our named entity recognizer and the extracted rules to build a translation model that encompassed word-embedding models. Though our transliteration module did not yield a result we anticipated for Tamil→Sinhala translation due to its phonological complexity, other parts contributed towards an increase in performance.

Overall, our hypothesis of increasing the performance of the existing translation system by incorporating named entity translation can use to improve parallel glossaries.

Chapter 7

Future Works

In general, the tag set we had proposed in general covers most of the named entities in official government documents. However, further analysis of the corpus revealed us that our approach would have benefited by introducing new tags to cover events, buildings and location. Using the knowledge we have gained from this research, as a future addition we can look into using a more fine-grained approach.

The current implementation of our GSSL approach uses Linear Discriminant Analysis (LDA). LDA calculations are done mostly in memory. Thus, when we attempt to use larger annotated training sets with each vector having over 900 dimensions, it leads to memory overflows. Since our target was towards addressing low-resource settings, we did not attempt to address this issue. As a part of future research, we can attempt different approaches like kNN-CUDA that uses GPUs to identify nearest neighbors. For the classification also we only experimented with a trivial label propagation algorithm. As part of future research, new classification algorithms akin to that done by Yang et al. [141] can be employed to see if they produce better results.

One of the key takeaways with regards to the deep-learning approach is that it works optimally well with large amounts of data. So, the next step towards increasing the existing performance is to increase the amount of annotated data

As far as translation is concerned, we only introduced named entities as an input pre-processing step. Since it resulted in a better performance, we stopped investigating further. As part of future research, we can look into incorporating it at a training level and see how it affects the final performance.

Appendix A: Graph Based Semi-Supervised Learning for Tamil POS Tagging

Mokanarangan Thayaparan, Surangika Ranathunga, Uthayasanker Thayasivam
Dept. of Computer Science and Engineering
University of Moratuwa, Katubedda 10400, Sri Lanka
{mokanarangan.11, surangika, rtuthaya }@cse.mrt.ac.lk

Abstract

Parts of Speech (POS) tagging is an important pre-requisite for various Natural Language Processing tasks. POS tagging is rather challenging for morphologically rich languages such as Tamil. Being low-resourced, Tamil does not have a large POS annotated corpus to build good quality POS taggers using supervised machine learning techniques. In order to gain the maximum out of the existing Tamil POS tagged corpora, we have developed a graph-based semi-supervised learning approach to classify unlabelled data by exploiting a small sized POS labelled data set. In this approach, both labelled and unlabelled data are converted to vectors using word embeddings and a weighted graph is constructed using Mahalanobis distance. Then semi-supervised learning (SSL) algorithms are used to classify the unlabelled data. We were able to gain an accuracy of 0.8743 over an accuracy of 0.7333 produced by a CRF tagger for the same limited size corpus.

Keywords: Semi-Supervised Learning, Low-resourced languages, Graph-based SSL, Word Embedding, POS tagging

1. Introduction

In the recent past, supervised learning methods have produced high accuracies for Parts-of-Speech (POS) tagging (Gimenez and Marquez, 2004). In particular, sequence models such as hidden Markov models (HMM) and conditional random fields (CRF) have given good results (Huang et al., 2015). However, these techniques rely on the availability of relatively large amounts of annotated data. Hence, building an accurate domain insensitive POS tagger is challenging for low resourced languages.

Tamil is one such low resourced language, which is widely used in South India and Sri Lanka. There have been several POS taggers developed for Tamil language using supervised learning techniques (Dhanalakshmi et al., 2009)(Pandian and Geetha, 2009). Since the annotated corpora used in this research have been of small size and from a single domain, these supervised techniques greatly suffer from accuracy and domain adaptability (Rani et al., 2016). For example, FIRE corpus (Forum for Information Retrieval Evaluation, 2014), a widely used freely available Tamil POS annotated corpus contains only 80k words. In contrast, the Wall Street corpus, which is an English POS-annotated corpus has a word count of 1,173K words (Gimenez and Marquez, 2004), meaning that the size of the FIRE corpus is approximately 15 times smaller than the Wall Street corpus. Thus, when using a small corpus such as FIRE, we cannot expect similar accuracy to that of English when supervised techniques are used. Moreover, these approaches depend on language dependent features such as morphological tags (Dhanalakshmi et al., 2009) thus limiting the scalability for adapting to other low resourced languages.

In contrast to supervised approaches, semi-supervised approaches such as graph based semi-supervised learning and manifold regularization (Niyogi, 2013) use both labeled and unlabelled data for their classification, and have proven to work with a small data sets (Zhu et al., 2003). Despite having smaller sized POS-tagged data for Tamil, there has been only two research leveraging the opportunity presented by semi-supervised learning. Ganesh et al. (2014) have used segmentation patterns to implement a bootstrapping approach for POS tagging. This approach relies on language dependent data such as suffix context patterns. Rani et al. (2016) use small annotated training data to build a classifier model using context-based association rule mining. This approach neither includes any language-specific linguistic information nor requires a large corpus. However, they collect all possible words occurring in the same context from the untagged data into a list called context-based list, thus limiting it from scaling to large monolingual corpus.

Graph based semi-supervised learning (SSL) has gained traction in Natural Language Processing tasks such as question answering (Celikyilmaz et al., 2009), structural tagging (Subramanya et al., 2010), and speech language recognition (Liu et al., 2016). Graph based SSL builds a meaningful graph using labelled and unlabelled instances. It then employs an SSL algorithm such as harmonic functions (Zhu et al., 2005) or label propagation (Zhu et al., 2003) to label the unlabelled instances. Graph based SSL is easily parallelizable and scalable to large data (Zhu et al., 2005).

In this paper, we present a novel graph-based semi-supervised approach to produce an accurate POS tagger for Tamil using a limited size corpus. Our idea is inspired by Talukdar and Pereira (2010)'s case study on modified absorption, which is a label propagation algorithm. They have implemented a Named Entity recognizer by building a connected word graph. Similarity between words is measured using WordNet. Then they employ label propagation to assign labels to all the unlabelled nodes.

Since Tamil is a low resourced language with no proper WordNet, we built a connected word graph using word vectors and employed label propagation. Our method is based on the clustering hypothesis that relative distance of word vectors of similar categories is lower than those between different categories. We use neural word embedding (Word2Vec (Mikolov et al., 2013), FastText (Joulin et al., 2016)) to create word vectors. Mahalanobis distance is used for measuring the distance (metric learning)

between these vectors in order to construct the graph. Mahalanobis distance generalizes the standard Euclidean distance, and has proven to be more effective (Davis et al., 2007). We empirically tested with four different metric learning algorithms (Information Theoretic Metric Learning (ITML) (Davis et al., 2007), Sparse Determinant Metric Learning (SDML) (Qi et al., 2009), Least Squares Metric Learning (LSML) (Liu et al., 2012), and Local Fisher Discriminant Analysis (LFDA) (Sugiyama, 2006)) to calculate Mahalanobis distance. Once the graph is constructed with labeled and unlabeled nodes, to assign labels to unlabeled nodes, we experimented with three different SSL algorithms (LP-ZGL) (Zhu et al., 2003), Absorption (Talukdar et al., 2008) and Modified Absorption (MAD) (Talukdar and Pereira, 2010)). Local Fisher Discriminant Analysis (LFDA) metric learning coupled with Label Propagation(LP-ZGL) yielded a maximum accuracy of 0.8743 for the FIRE corpus against a baseline accuracy of 0.7338 achieved by using a traditional CRF model. Unlike supervised learning approaches, our approach does not require a large high quality annotated data set, or language dependent features.

Thus the contributions of this paper are: (1) converting words to vectors using neural word embedding and building meaningful word graphs, (2) using Mahalanobis distance to measure relationships between word vectors, hence measuring the correlation between variables, and (3) using a language independent graph based semi-supervised approach for POS tagging in Tamil.

The rest of the paper is organized as follows. Section 2 discusses graph based semi supervised learning techniques and previous attempts on Tamil POS tagging. Section 3 details the data set used in our experiment. Section 4 discusses the methodology and how we implemented the system. Section 5 details the experiments carried out and the relevant results. Section 6 and Section 7 document the conclusion and future work, respectively.

2. Related Work

2.1. Graph based Semi-supervised Learning

Graph theory and Natural Language Processing are well studied disciplines, but are commonly perceived as distinct with different algorithms and with different applications. But recent research has shown that these disciplines are connected and graph-theoretical approaches can be employed to find efficient solutions for NLP problems. Entities are connected by a range of relations in many NLP problems and graph is a natural way to capture the relationship between the entities. Graph based approaches have been used in word sense disambiguation, entity disambiguation, thesaurus construction, textual entailment and semantic classification (Mihalcea and Radev, 2011).

Graph based semi-supervised learning builds graphs connecting labeled and unlabeled data points, and perform classification by propagating the labels. The graph is constructed to reflect our prior knowledge about the domain. The intuition is that similar data points have similar labels. We let the hidden/observed labels be random variables on the nodes of this graph. Labels are injected to unlabeled nodes from labeled nodes. Graphs provide a uniform representation for heterogeneous data and are easily parallelizable (Zhu et al., 2005).

One of the challenges of graph based approach is building the graph that reflects the relationship between entities. Depending on the task, the nodes and edges may represent a variety of language related units and links. Different NLP tasks have approached this challenge in different ways. For the task of opinion summarization, Zhu et al. (2013) constructed a graph of sentences linked by edges whose weight combines the term similarity and objective orientation similarity. And to perform discourse analysis in chat, Elsner and Charniak (2010) predicted the probabilities for pair of utterance as belonging the same conversation thread or not based on lexical, timing and discourse-based features. Then constructed a graph with each nodes representing the utterances and the edges representing the probability score between the nodes. Although these approaches are evidences for the versatility of graph based approaches, these cannot be adopted to a word level problem like sequential tagging. Using graph methods for sequential tagging relies on the belief that similar words will have the same tag. Unlike the aforementioned approaches, here the nodes in these graph represents words or phrases and the the edges will indicate the similarity between nodes. Talukdar and Pereira (2010) tag words with NER information through a label propagation algorithm on a word similarity graph built using WordNet information. Words are represented are the graph vertices and the edge denotes the WordNet relationship. This approach cannot be adopted for a low resource language which doesn't have a proper WordNet. Subramanya et al. (2010) POS tags on a similarity graph where local sequence contexts (n-grams) are vertices. The similarity function between graphs is the cosine distance between the point-wise mutual information vectors (PMI) representing each node. The point-wise mutual information is calculated between n-gram and set of context features. These context features includes suffixes, left word and right word contexts. The challenge of this approach is the scalability for a morphologically complex language like Tamil.

2.2. Tamil POS tagging

Tamil is a low resourced, morphologically rich language with many inflections and a complex grammatical structure. Thus, automatic POS tagging for Tamil is a challenging task. Supervised learning approaches have been heavily undertaken in Tamil for POS tagging. These include CRF models using morphological information (Pandian and Geetha, 2009) and Support Vector Machines (SVM) using semantic information (Dhanalakshmi et al., 2009). These models had been trained using different corpora containing approximately 200k annotated words. These annotated corpora or taggers are not publicly available.

There have been very few attempts in using semi-supervised approaches for Tamil language to develop POS taggers. Ganesh et al. (2014) have used language features with a bootstrapping approach to obtain a precision of 86.74%. They have presented a pattern based bootstrapping approach using only a small set of POS labelled suffix context patterns. The patterns consist of a stem and a sequence

of suffixes, obtained by segmentation using a manually created suffix list. This bootstrapping technique generates new patterns by iteratively masking suffixes with low probability of occurrences in the suffix context, and replacing them with other co-occurring suffixes. This approach relies on language specific information.
Rani et al. (2016) have employed a semi-supervised rule mining approach using morphological features for Hindi, Tamil, and Telugu languages. They have used a combination of a small annotated and untagged training data to build a classifier model using a concept of context-based association rule mining. These association rules work as context-based tagging rules.

3. Data set

For our experiment, we used the FIRE Tamil Corpus. The FIRE Tamil corpus contains 80k POS tagged words with 21 different tags as shown in Table 1.

NN	Noun
NNC	Compound Noun
RB	Adverb
VM	Verb Main
SYM	Symbol
PRP	Personal Pronoun
JJ	Adjective
NNP	Pronoun
PSP	Prepositions
QC	Quantity Count
VAUX	Verb Auxiliary
DEM	Determiners
QF	Quantifiers
NEG	Negatives
QO	Quantity Order
WQ	Word Question
INTF	Intensifier
NNPC	Compound Pro Noun
CC	Coordinating Conjunction
RBP	Adverb Phrase

Table 1: POS tagsets for FIRE Tamil Corpus

4. Methodology

Our work is inspired by Talukdar and Pereira (2010)'s case study on the performance of different algorithms for classification in graphs. In this work, words are represented as nodes and the similarity between nodes are measured using WordNet distance. Since Tamil is a low resourced language, this approach was not viable for us. Another approach was to represent words by converting them to vectors and computing the similarity. Subramanya et al. (2010) had employed a point wise mutual information (PMI) based approach to convert the word to vectors and compute the similarity by measuring the cosine distance. His approach used hand-crafted features that will not work with same efficiency across different languages.
Hence, an efficient way of representing a word in the vector space has to be determined. In addition, it is required to identify mechanisms for (1) constructing a meaningful graph based on the word vector, and (2) classifying unlabelled words based on the constructed graph by measuring the similarity.

4.1. Representing a word in the vector space

We adopted the Word2Vec model proposed by Mikolov et al. (2013) and convert the word into the vector space to construct the graph. To the best of our knowledge, Word2Vec has never been used to construct weighted word graphs to be used in SSL. Similarly we also experimented with Fast Text skipgram (Bojanowski et al., 2016) and bag of words models (Joulin et al., 2016). The key difference between Word2Vec and FastText is that Word2Vec treats each word in corpus as an atomic entity and generates a vector for each word. In contrast, FastText treats each word as composed of ngrams and the vector word is made of the sum of these vectors.

4.2. Constructing a meaningful graph based on the word vector

Each word is converted to a d dimensional vector space. Out of the n words in the list, n_l are labelled($n >>> n_l$). We employ 32 different tags to denote each POS entity (Dhanalakshmi et al., 2009). $G = (V, E, W)$ is the graph we are interested in constructing; where V is the set of vertices with $|V| = n$, E is the set of edges. W is the symmetric $n \times n$ matrix of edge weights we want to learn. Usually we could choose a standard distance metric (Euclidean, City-Block, Cosine, etc.). Instead, Mahalanobis distance has proven to be effective with clustering problems over the standard metrics (De Maesschalck et al., 2000).
We use a supervised method for learning the Mahalanobis distance. For this purpose, we need to calculate the positive definite matrix A of size $d \times n$ that parametrizes the Mahalanobis distance, $d_A(x_i, x_j)$ (Dhillon et al., 2010; Davis et al., 2007; Sugiyama, 2006) between words x_i and x_j as shown in Equation (1).

$$d_A(x_i, x_j) = (x_i - xj)^T A(x_i - x_j) \tag{1}$$

Since A is positive definite, it can be decomposed into $P^T P$, where P is another matrix of size $d \times d$

$$\begin{aligned} d_A(x_i, x_j) &= (x_i - x_j)^T P^T P(x_i - x_j) \\ &= (Px_i - Px_j)^T (Px_i - Px_j) \\ &= d_I(Px_i, Px_j) \end{aligned} \tag{2}$$

There are many proposed methods for calculating the transformation matrix P. We empirically experimented with different metric learning algorithms, including Information Theoretic Metric Learning (ITML) (Davis et al., 2007), Sparse Determinant Metric Learning (SDML) (Qi et al., 2009), Least Squares Metric Learning (LSML) (Liu et al., 2012), and Local Fisher Discriminant Analysis (LFDA) (Sugiyama, 2006).Researches in link prediction in networks (Shaw et al., 2011), music recommendation (McFee et al., 2011) and bio metrics verification (Ben et al., 2012) has shown that metric learning plays a vital role increasing accuracy of the system.

ITML minimizes the differential entropy between multivariate Gaussian under constraints on the distance function. Davis et al. (2007) have expressed the problem as that of minimizing the LogDet divergence subject to linear constraints. SDML uses l_1-penalized log-determinant regularization to calculate the metric. This algorithm exploits the sparsity nature underlying the intrinsic high dimensional feature space. LSML uses an algorithm that minimizes a convex objective function corresponding to the sum of squared residuals of constraints. Finally LFDA, is a linear supervised dimensionality reduction method which is particularly useful when dealing with cases where one or more core classes consist of separate clusters in input space.

We calculate P using each of these metric learning algorithms and project the words into a new space to calculate Px_i. Based on Equation 2, we compute the Euclidean distance in the linearly transformed matrix. Gaussian kernel $[2, 16]$ was used to compute the similarity between words as shown in Equation 3 (Dhillon et al., 2010). We then sparsify the graph by selecting k neighbors for each node and set the weights to zero for all others (Zhu et al., 2003).

$$W_{ij} = exp(\frac{-d_A(x_i, x_j)}{2\sigma^2}) \quad (3)$$

The culmination of all these steps results in a meaningful graph where relative distances of word vectors of similar categories will be lower than those between different categories.

4.3. Classifying Unlabelled Nodes based on the Constructed Graph

Once the graph is constructed, unlabelled words in the graph should be classified. For this, we experimented with Label Propagation(LP-ZGL), and Absorption and Modified Absorption (MAD) techniques. LP-ZGL (Zhu et al., 2003) was one of the first graph based SSL methods. LP-ZGL propagates the labels over the graph by penalizing any label assignment where two nodes connected by a highly weighted edge are assigned different labels. LP-ZGL prefers smooth labeling over the graph. This property is also shared by the other two algorithms. Absorption (Talukdar et al., 2008) has been used for open domain class-instance acquisition. Absorption is an iterative algorithm where label estimates depend on the previous iteration. Modified Absorption (MAD) (Talukdar and Pereira, 2010) shares the same properties of the Absorption algorithm but can be expressed as an unconstrained optimization problem. We experimented with all these algorithms to estimate the labels of the untagged words.

5. Experiments and Results

5.1. Experiments

We split the data into 60k words for training and 20k words for testing. To the best of our knowledge, there has been only Named Entity Recognition research (Abinaya et al., 2014) done in Tamil using FIRE corpus and no POS tagging research done.

We trained both Word2Vec and FastText models with a word window of three (the commonly used window size) using the Tamil Wikipedia corpus (Wikipedia, 2016) (about 1M words) after removing only the punctuation marks. We used these models to convert word to vector form. Each vector is of 300 dimensions. For graph construction, a subset of 3000 sentences with approximately 50k unlabelled words from the Tamil Wikipedia corpus were added to the set. We constructed the word graphs using the aforementioned four metric learning approaches and employed three labeled propagation approaches to identify the best combination.

Since most of the successful approaches related to Tamil POS tagging have been carried out using Conditional Random Fields (CRF) (Pandian and Geetha, 2009), we used the same approach with word trigram feature as our baseline method. Here, trigrams were selected because for Word2Vec and FastText models also, a word window of three was used.

5.2. Results

The following Tables 2-5 document the results obtained for each graph construction algorithm in combination with the classification methods.

Word To Vector Algorithm	MAD	Abs	LP-ZGL
Word2Vec (SkipGram)	0.7534	0.7531	0.7201
Word2Vec (Bag of words)	0.6945	0.6967	0.6754
Fasttext (SkipGram)	0.8146	0.814	0.822
Fasttext (Bag of Words)	0.795	0.7952	0.801

Table 2: Accuracy of Information Theoretic Metric Learning

Word To Vector Algorithm	MAD	Abs	LP-ZGL
Word2Vec (SkipGram)	0.7012	0.701	0.721
Word2Vec (Bag of words)	0.6641	0.6542	0.665
Fasttext (SkipGram)	0.7886	0.7935	0.7988
Fasttext (Bag of Words)	0.7712	0.775	0.7767

Table 3: Accuracy of Sparse Determinant Metric Learning

Word To Vector Algorithm	MAD	Abs	LP-ZGL
Word2Vec (SkipGram)	0.734	0.733	0.732
Word2Vec (Bag of words)	0.701	0.71	0.711
Fasttext (SkipGram)	0.8547	0.861	0.8634
Fasttext (Bag of Words)	0.823	0.834	0.845

Table 4: Accuracy of Least Squares Metric Learning

Word To Vector Algorithm	MAD	Abs	LP-ZGL
Word2Vec (SkipGram)	0.7678	0.7775	0.7757
Word2Vec (Bag of words)	0.7664	0.7567	0.7456
Fasttext (SkipGram)	0.8673	0.8573	**0.8743**
Fasttext (Bag of Words)	0.85	0.853	0.86

Table 5: Accuracy of Local Fisher Discriminant Analysis

As illustrated above, Local Fisher Discriminant Analysis(LFDA) combined with Label propagation yields the best accuracy of 0.8743. LFDA is a linear supervised dimensionality reduction method. It proved effective in our case since each of our words had a size of 300 dimensions. FastText(skipgram) in combination with label propagation consistently performed better than other algorithms in all graph construction methodologies.

To test the robustness of the approach, we trained the best performing combination (LFDA and LP-ZGL) with 20k words and tested with 60k words. It yielded an accuracy of 0.753. Meanwhile, the baseline CRF model only gave an accuracy score of 0.633. This proves that our approach is more robust even when the labelled data set is comparatively small.

6. Conclusion

Our research establishes the fact that graph based semi-supervised approaches are more robust than supervised classification algorithms for POS tagging when the data set is relatively small. Thus graph based semi supervised data can be employed in the early stages of creating POS tagged data sets. Human annotators can correct the automatically annotated corpus with less effort, and the corrected annotated data set can be used in an iterative manner to re-train the tagger. Thus, graph based semi-supervised approaches are particularly useful for POS tagging of low-resourced languages such as Tamil. We used neural word embedding to create a vector representation of words, and Mahanalobis distance to measure distance between word vectors in order to build the graph. This shows that word embedding provides an excellent alternative for WordNet in measuring similarity between words, especially for languages that do not have a WordNet. This is useful not only for graph building, but for any task that requires measuring the similarity of words.

7. Future work

Our language independent work has shown promise with low resources. We have only done the research for one language, and this research should be extended to other languages to verify the general applicability of the presented methodology. We hope to extend this idea for other low resourced sequential tagging problems such as Named Entity Recognition. This research can also be extended to improve and incorporate other word embedding techniques such as VarEmbed that uses morphological priors for probabilistic neural word embedding (Bhatia et al., 2016). We can also experiment with other graph construction algorithms such as b-matching (Jebara et al., 2009). The main limitation of this technique is the amount of time taken to build the graph. Thus we intend to look into different code optimization methods. While we have compared our approach with the pure CRF implementation, Lample et al. (2016) has shown that CRF in combination with LSTM can provide a higher accuracy for Named entity recognition but that approach has not been tried for POS tagging in morphologically complex languages such as Tamil. We are eager to see how our approach stacks up with them.

8. Acknowledgement

This research is partially funded by the National Languages Processing Centre of University of Moratuwa, Sri Lanka. The authors would also like to thank the LKDomain registry for partially funding this publication.

9. Bibliographical References

Abinaya, N., John, N., Ganesh, B. H., Kumar, A. M., and Soman, K. (2014). Amrita_cen@ fire-2014: Named entity recognition for indian languages using rich features. In *Proceedings of the Forum for Information Retrieval Evaluation*, pages 103–111. ACM.

Ben, X., Meng, W., Yan, R., and Wang, K. (2012). An improved biometrics technique based on metric learning approach. *Neurocomputing*, 97:44 – 51.

Bhatia, P., Guthrie, R., and Eisenstein, J. (2016). Morphological priors for probabilistic neural word embeddings. 3 August.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Celikyilmaz, A., Thint, M., and Huang, Z. (2009). A graph-based semi-supervised learning for question-answering. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 719–727, Stroudsburg, PA, USA. Association for Computational Linguistics.

Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 209–216, New York, NY, USA. ACM.

De Maesschalck, R., Jouan-Rimbaud, D., and Massart, D. L. (2000). The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18.

Dhanalakshmi, V., Rajendran, S., Soman, K. P., and Edu, K. (2009). POS tagger and chunker for tamil language.

Dhillon, P. S., Talukdar, P. P., and Crammer, K. (2010). Inference driven metric learning (idml) for graph construction.

Elsner, M. and Charniak, E. (2010). Disentangling chat. *Comput. Linguist.*, 36(3):389–409, September.

Ganesh, J., Parthasarathi, R., Geetha, T. V., and Balaji, J. (2014). Pattern based bootstrapping technique for tamil POS tagging. In *Mining Intelligence and Knowledge Exploration*, Lecture Notes in Computer Science, pages 256–267. Springer, Cham.

Gimenez, J. and Marquez, L. (2004). Svmtool: A general pos tagger generator based on support vector machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Jebara, T., Wang, J., and Chang, S.-F. (2009). Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 441–448, New York, NY, USA. ACM.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.

Liu, E. Y., Guo, Z., Zhang, X., Jojic, V., and Wang, W. (2012). Metric learning from relative comparisons by minimizing squared residual. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, pages 978–983, Washington, DC, USA. IEEE Computer Society.

Liu, Y., Kirchhoff, K., Liu, Y., and Kirchhoff, K. (2016). Graph-based semisupervised learning for acoustic modeling in automatic speech recognition. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 24(11):1946–1956, November.

McFee, B., Barrington, L., and Lanckriet, G. R. G. (2011). Learning content similarity for music recommendation. *CoRR*, abs/1105.2344.

Mihalcea, R. F. and Radev, D. R. (2011). *Graph-based Natural Language Processing and Information Retrieval*. Cambridge University Press, New York, NY, USA, 1st edition.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. 16 January.

Niyogi, P. (2013). Manifold regularization and semi-supervised learning: Some theoretical analyses. *Journal of Machine Learning Research*, 14:1229–1250.

Pandian, S. L. and Geetha, T. V. (2009). Crf models for tamil part of speech tagging and chunking. In *Proceedinghs of the 22Nd International Conference on Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy*, ICCPOL '09, pages 11–22, Berlin, Heidelberg. Springer-Verlag.

Qi, G.-J., Tang, J., Zha, Z.-J., Chua, T.-S., and Zhang, H.-J. (2009). An efficient sparse metric learning in high-dimensional space via l1-penalized log-determinant regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 841–848, New York, NY, USA. ACM.

Rani, P., Pudi, V., and Sharma, D. M. (2016). A semi-supervised associative classification method for POS tagging. *Int J Data Sci Anal*, 1(2):123–136, 1 July.

Shaw, B., Huang, B., and Jebara, T. (2011). Learning a distance metric from a network. In J. Shawe-Taylor, et al., editors, *Advances in Neural Information Processing Systems 24*, pages 1899–1907. Curran Associates, Inc.

Subramanya, A., Petrov, S., and Pereira, F. (2010). Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 167–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sugiyama, M. (2006). Local fisher discriminant analysis for supervised dimensionality reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 905–912, New York, NY, USA. ACM.

Talukdar, P. P. and Pereira, F. (2010). Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1473–1481, Stroudsburg, PA, USA. Association for Computational Linguistics.

Talukdar, P. P., Reisinger, J., Paşca, M., Ravichandran, D., Bhagat, R., and Pereira, F. (2008). Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 582–590, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 912–919. AAAI Press.

Zhu, X., Lafferty, J., and Rosenfeld, R. (2005). *Semi-supervised learning with graphs*. Ph.D. thesis, Carnegie Mellon University, language technologies institute, school of computer science.

Zhu, L., Gao, S., Pan, S. J., Li, H., Deng, D., and Shahabi, C. (2013). Graph-based informative-sentence selection for opinion summarization. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 408–412, New York, NY, USA. ACM.

10. Language Resource References

Forum for Information Retrieval Evaluation. (2014). *FIRE Corpus*. Indian Institute of Science, Bangalore.

Wikipedia. (2016). *Tamil Wikipedia Corpus*. Wikipedia.

Appendix B: Configuration File Used to Build CRF Tagger with AllenNLP

```
{
   "dataset_reader":{
      "type":"sequence_tagging",
      "word_tag_delimiter":"\t",
      "token_delimiter":"\n",
      "token_indexers":{
         "tokens":{
            "type":"single_id"
         },
         "elmo":{
            "type":"elmo_characters"
         },
         "token_characters":{
            "type":"characters"
         }
      }
   },
   "train_data_path":"/src/data/Tamil_NER_Clean/train.txt
   ",
   "validation_data_path":"/src/data/Tamil_NER_Clean/dev.
   txt",
   "test_data_path":"/src/data/Tamil_NER_Clean/test.txt",
   "evaluate_on_test":true,
   "model":{
      "type":"crf_tagger",
      "text_field_embedder":{
```

```
        "tokens":{
            "type":"embedding",
            "embedding_dim":300,
            "pretrained_file":"/src/vectors/Wang_Tamil.
txt.gz"
        },
        "elmo":{
            "type":"elmo_token_embedder",
            "options_file":"/src/options.json",
            "weight_file":"/src/vectors/tamil_elmo.hdf5",
            "do_layer_norm":false,
            "dropout":0.5
        },
        "token_characters":{
            "type":"character_encoding",
            "embedding":{
                "embedding_dim":25
            },
            "encoder":{
                "type":"gru",
                "input_size":25,
                "hidden_size":80,
                "num_layers":2,
                "dropout":0.25,
                "bidirectional":true
            }
        }
    },
    "encoder":{
        "type":"gru",
```

```
            "input_size":1484,
            "hidden_size":300,
            "num_layers":2,
            "dropout":0.25,
            "bidirectional":true
        },
        "regularizer":[
            [
                "transitions",
                {
                    "type":"l2",
                    "alpha":0.01
                }
            ]
        ]
    },
    "iterator":{
        "type":"basic",
        "batch_size":32
    },
    "trainer":{
        "optimizer":"adam",
        "num_epochs":20,
        "patience":10,
        "cuda_device":-1
    }
}
```

Listing 1: CRF tagger configuration

Appendix C: Graph based semi-supervised sequence tagging for low resourced languages

Anonymous EMNLP submission

Abstract

We present a novel Graph-based Semi-Supervised Learning (GSSL) approach for sequence tagging tasks. Performance gains over traditional GSSL techniques are achieved by capturing local context information in graph representation, as well as by producing a low-dimensional graph representation that separates nodes belonging to distinct categories. This GSSL approach far outperforms the other state-of-the-art techniques in low-resourced settings, thus proving to a viable solution for sequence tagging for low-resourced languages.

1 Introduction

When supervised data is scarce, it has been common to employ semi-supervised learning (SSL) techniques for many different Natural Language Processing (NLP) tasks (Garrette et al., 2013; Cheng et al., 2016). In general, graph-based semi-supervised learning (GSSL) techniques have shown even better performance than other SSL techniques (Subramanya and Bilmes, 2008). Graphs of words capture term dependence, encode the strength of the dependence as edge weights, and capture term order (via directed edges) (Rousseau and Vazirgiannis, 2013; Skianis et al., 2016; Rousseau et al., 2015). Hence, GSSL shows greater potential for NLP tasks. They have been used in word sense disambiguation, entity disambiguation, thesaurus construction, textual entailment, and semantic classification (Mihalcea and Radev, 2011), which suggests that semantic relationships between words have been exploited in graph construction.

As for sequence tagging, there are two key factors in constructing a meaningful graph. First, it is important to be able to represent each word occurrence (token) as a vertex because the label assignment for the same word type may differ based on the context it is used. Second it is important to link vertices that are likely to have the same label, where edge weights govern how strongly the labels of the nodes linked by the edge should agree. Given such a graph, a label propagation algorithm could label the unlabeled vertices based on the information of their nearest neighbours.

Related literature suggests that *types* have been the common choice for representing vertices in the graph (Mihalcea and Tarau, 2004). Early work on using GSSL for sequence tagging problems also relied on this word-based representation (Talukdar and Pereira, 2010), thus missing out context information in their vertex representation. These approaches mostly rely on word based similarity measures to determine edge weights.

The alternative way to represent vertices is using local sequence contexts (n-gram). A notable work along this line was reported by Subramanya et al. (2010), which exploited the empirical observation that the Parts of Speech (POS) of a word occurrence is mostly determined by its local context. They represent each vertex using a vector of pointwise mutual information (PMI) values, computed using the n-gram and each of the features that occur with tokens of that n-gram. The cosine distance between these PMI vectors of a pair of vertices is used as edge weights between those vertices.

Instead of these PMI-based count models, much recent GSSL work for sequence tagging reported the use of traditional neural word embeddings such as WORD2VEC (predict models) for representing vertices of the graph (Mokanarangan et al., 2018; Demirel, 2017). These predict models are much more concise than PMI vectors. However, these traditional WORD2VEC approaches are less sensitive to word order (the local context of a word occurrence), which makes them sub-optimal for sequential learning problems (Ling et al., 2015).

There is another limitation of these approaches, which is not necessarily limited to GSSL methods that use predict models, but applicable for any GSSL method. The foundation assumption in GSSL is that the similar nodes will carry same labels. Even though this assumption is effective in many cases, this is not completely true for many sequence labeling problem instances. For example, the word *'amazed'* and the word *'fantastic'* are semantically very similar but they should be labeled with different POS tags.

We present a novel graph building approach to tackle the above limitation of count models used in GSSL techniques for sequential tagging. We adopt the graph building methodology mentioned in Mokanarangan et al. (2018), but leverage the structured embedding models presented by Ling et al. (2015) and Peters et al. (2017), which are more sensitive to word order. We empirically evaluate some compelling choices for aggregating these n-gram token vectors to represent n-grams effectively.

In order to tackle the second limitation, a graph constructed using this n-gram representation is transformed into a lower dimensional vector space in such a way that vertices belonging to different classes are well-separated. This helps to reduce overall computational complexity as well.

We evaluate our approach for three different sequence tasks (POS, Named Entity Recognition (NER), and Chunking) for English using the CoNLL 2003 data set (Tjong Kim Sang and Buchholz, 2000), and for POS for Sinhala and Tamil. For each experiment, we use 1 million unlabeled tokens. We vary the amount of labelled tokens in a step-wise manner until up to 100,000 tokens, to resemble a low-resourced setting. Results show that our solution outperforms the state-of-the-art techniques for sequence tagging when the amount of training data is less than 80,000 tokens.

2 Related Work

Early work on using GSSL for sequence tagging problems relied on word-based graph representations. Talukdar and Pereira (2010) had constructed a word graph using WordNet to perform NER. In this approach, vertices are noted as surface level word forms and each relationship in WordNet is represented as an edge. Although simple and straightforward, this approach fails to capture the syntactic information essential for sequence classification tasks.

In contrast, Subramanya et al. (2010) represent each vertex using a vector of point-wise mutual information (PMI) values, computed using the n-gram and each of the features that occur with tokens of that n-gram. The cosine distance between these PMI vectors of a pair of vertices are used as edge weights between those vertices. These PMI vectors are capable of capturing local context information. However, they note that the vectors used in this approach are sparse and high dimensional.

Extending on Subramanya et al. (2010)'s work, Das and Petrov (2011) designed unsupervised POS taggers for languages that have no labeled training data. They constructed a graph based on the same PMI features introduced by Subramanya et al. (2010), and used graph-based label propagation for cross-lingual knowledge transfer. This solution was based on the observation that despite the language differences, words in different languages share similar relationships in local context.

In their research on graph-based posterior regularization for semi-supervised structured prediction, He et al. (2013) claimed that using Subramanya et al. (2010)'s features to build graphs leads to unrelated trigrams to match. Instead they proposed a different set of features to build PMI based graphs which also suffers from sparsity.

Recently, Demirel (2017) had proposed an approach to solve POS tagging where every word in a corpus is connected into a graph where each node is denoted by a word embedding vector. They capture the word ordering information by connecting each word to next and previous word in the corpus. This graph is then directly fed into a neural network model called graph convolutional network (GCN) for classification.

Exploiting the cluster assumption of word embedding, Mokanarangan et al. (2018) had proposed an approach where each node is represented by a word embedding vector, and edges between nodes are calculated using supervised metric learning. Though this approach has shown promise in low resourced settings, it fails to capture different context information for the same word.

3 Graph Construction and Label Propagation

3.1 Representing Nodes of the Graph

In sequence tagging problems, label of a word is predominantly determined by its context. Thus, syntactic relationships between word tokens play a major role. For example, the word *present* may appear as a noun or a verb, depending on the context. Thus, without referring to the context, the exact POS tag of the word cannot be determined. As an example with respect to Named Entities (NEs), consider the NEs "Central Bank spokesman" and "The Central African Republic". Here, the word 'Central' is used as part of both an Organization and Location (Peters et al., 2017).

As opposed to using lexical units or simple word vector representations to create nodes, we experiment with different types of vector representations.

Related literature presents contradicting arguments with respect to the performance of count models and predict models. Baroni et al. (2014) and Mikolov et al. (2013) claim that predict models such as WORD2VEC and FASTTEXT capture more syntactic and semantic information compared to traditional count based distributional models such as PMI vectors. However, much recently Levy et al. (2015) have claimed that with proper system choices and hyper parameters, traditional count models can yield similar gains. However, in count models, increasing the unlabeled data produces extremely spares vectors that leads to computationally demanding graph building. Thus we experimented with the following predict models that have claimed to capture syntactic information.

WANG2VEC (Ling et al., 2015): WANG2VEC is presented as a model that captures more syntactic-oriented embedding than WORD2VEC. Though this still produces same vector representations for words in different contexts, experiments have shown that vectors produced are syntactically close.

FASTTEXT (Bojanowski et al., 2016): While WORD2VEC treats each word in corpus as an atomic entity and generates a vector for each word, FASTTEXT treats each word as comprised of n-grams and the vector is made of sum of these vectors. Previous research (Mokanarangan et al., 2018) has shown that FASTTEXT performs well when compared with WORD2VEC in GSSL settings.

ELMO (Peters et al., 2017): This semi-supervised bidirectional language model computes an encoding of the context at each position in the sequence. It has been proved that ELMO surpasses the state of the art approaches in capturing semantic and syntactic models. Although rich with information, it is computationally exhaustive to create these vectors. Unlike other word embedding models used, this model produces vectors for a word based on the contextual information of the word.

As mentioned earlier, we base our work on one assumption that words with same local sequence context will have the same sequence tags. In order to capture the local context information in our graph, we experimented with one solution: concatenation of vector n-grams.

3.2 Creating Edges of the Graph

Similar to the approach proposed by Subramanya et al. (2010), once the nodes in the graph are fixed, the edge weights w_{ij} between them between two vector n-grams i and j are defined as shown in Equation 1.

$$w_{ij} = \begin{cases} sim(i,j), & \text{if } i \in K(j) \text{ or } j \in K(i). \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Here $K(i)$ in the set of k-nearest neighbors of vector n-gram i. The similarity function was defined using the Gaussian kernel denoted in Equation 2 (Dhillon et al., 2010). Here $d(x_i, x_j)$ is the euclidean distance between vectors i and j.

$$sim(i,j) = exp(\frac{-d(x_i, x_j)}{2\sigma^2}) \quad (2)$$

Theoretically, there can be an edge between each pair of nodes in the graph. However, one can safely disregard edges that have very low weights, because the relationship between such nodes is very weak. Such weak edges can add noise to label propagation.

The identification of the set of vertices that should be connected to a given vertex can be modelled in the form of k-nearest neighbour problem, where the objective is to determine the set of vertices that have the strongest relationship with the given node (i.e., we determine the set of edges with the highest weight for a given node). Determining the set of edges using k-nn is more effec-

tive if the vertices belonging to different classes are well-separated. Thus we transform the vector space into a lower dimension while preserving the separation of classes.

This dimensionality reduction serves another purpose. The performance of nearest neighbor algorithms degrades when the size of the vector increases. Since we used word embedding models result in 300 dimensions. When concatenating vector n-grams, this dimension reaches 900. Thus the dimensionality reduction makes graph construction extremely efficient.

Algorithm 1 presents the graph construction procedure.

Algorithm 1: GSSL using word embedding

Data: Corpus with n number of words where n_l are labeled ($n >>> n_l$)
for *each* w_i *in corpus* **do**
  $vec_i = ConvertWordToVector(w_i)$;
  $v_i = Concatenate(vec_{i-1}, vec_i, vec_{i+1})$;
end
$V_r = BuildVectorList(v)$;
$V_s = SupervisedReduction(V_r)$;
for *each* v_i *in* V **do**
  $e_i = NearestKVectors(v_i, distance =' euclidean')$;
  $w_i = CalculateWeight(e_i)$
end
$E = BuildEdgeMatrix(e)$;
$W = BuildWeightMatrix(w)$;
Build graph $G = (V, E, W)$;
$Predict(G, n)$

3.3 Label Propagation

Label propagation refers to the process of assigning labels to unlabeled nodes using the labelled nodes. The prior assumption of semi-supervised learning is that nearby points and points on the same structure are likely to have the same labels (Zhu et al., 2003). This is a simple and straightforward approach that have been the staple of semi-supervised learning and have yielded encouraging results.

4 Implementation

As mentioned above, high dimensionality of the vectors and the large size of the sample space severely affect the performance of k-nn algorithm. Thus we resorted to approximate nearest neighbor algorithms(ANN). We use Annoy (Bernhardsson, 2018), which has been empirically shown to work better with large data-sets (Aumüller et al., 2017). k was set to an arbitrary value of 20. It should be noted this ANN's accuracy drops when dimensions of the vector is greater than 100. This attribute played an important role in choosing to reduce dimensions.

To achieve a discriminant feature set in a lower dimension, two dimensionality reduction techniques were experimented with Linear discriminant analysis (LDA) and Fisher linear discriminant analysis (LFDA). Both LDA and LFDA are supervised methods that are useful in finding dimensions which aim at separating the clusters (Sugiyama, 2006).

For label propagation, Harmonic Function (HMN) (Zhu et al., 2003) and Local and Global Consistency (LGC) (Zhou et al., 2003) were experimented with. These are two of the well-established label propagation algorithms that have proven their effectiveness in different contexts (Zhu, 2005).

5 Experiments and Results

5.1 Data set

English. We evaluated our approach on CoNLL2003 NER task (Sang and Meulder, 2003) for POS, NER and Chunking task. We emulated a low resource setting for English by using only 20K, 40K, 60K and 100K data as our training setting as opposed to using the full training data.

Tamil. Tamil belongs to the Dravidian language family, which is used in some parts of South Asia. For Tamil we used the dataset from the Forum for Information Retrieval (FIRE) (Majumder et al., 2008). The dataset has nearly 80K labeled data with 32 POS classes.

Sinhala. Sinhala is an Indo Aryan language predominantly used in Sri Lanka. It has evolved from the same language family as Hindi, but being a language limited to an island nation, it has evolved to have its own characteristics. Sinhala is an ideal example of a low-resourced language. For our experiments, we used the University of Moratuwa (UOM) Sinhala POS corpus (Fernando et al., 2016), which currently has 260K tagged tokens labeled using 32 tags.

5.2 Experiment Setup

Experiments are designed to determine the impact of local context information in graph construction for sequence tagging tasks, and the impact of dimensionality reduction on the same. For English, we test the performance of our solution with respect to POS tagging, NER, and Chunking tasks of the CoNLL 2003 dataset. With respect to Tamil and Sinhala, we experiment only with POS tagging, due to the unavailability of data for other tasks.

The current implementation employs the Continuous Bag of Words (skip-gram) model of FASTTEXT (Bojanowski et al., 2016) to generate word embeddings for English, where the vector dimension is 300.

WANG2VEC models are generated using a part of the wiki dump for all the three languages. Dimension of these vectors is also set to 300.

ELMO model (Peters et al., 2017) of 1024 dimensions was reused. ELMO model was not used for Sinhala and Tamil, since we do not have enough computer capacity required to generate the model.

We have experimented with n = 3, when generating vector n-grams. For example when n = 3, in the example given in Section 3, the word "Central" will be represented by concatenating the word vectors of "The", "Central", "African", thus adding the context information. Thus we end up with a feature vector of 900 dimensions for FASTTEXT and WANG2VEC, and 3072 for ELMO.

For each language, the graph is constructed using 1 million tokens from an unlabeled corpus, and the labeled text size is varied from 20k to 100k in a step-wise manner.

To show that our GSSL solution works in low-resourced settings better than the state-of-the-art reported in the context of high-resourced settings, we compare our results with the work of Peters et al. (2017). We sampled the same amount of training samples from the CoNLL 2003 Shared Task (Sang and Meulder, 2003). For this experiment, according to the discussion by Peters et al. (2017), we used two bidirectional GRUs with 80 hidden units and 25 dimensional character embeddings for the token character encoder. The sequence layer uses two bidirectional GRUs with 300 hidden units each. For regularization, we add 25% dropout to the input of each GRU, but not to the recurrent connections to setup the model. We also embed the ELMO model to represent each word in this bidirectional model and tested it.

5.3 Results

For POS we report the accuracy, while for Chunking and NER we report the official evaluation metric (micro-averaged F1 score).

Both LDA and LFDA showed near equal performance, and so did HMN and LGC. Thus the following results only showcase the experiment setups that used LDA and HMN.

Table 1 shows the impact of different word embedding models in vertex representation, with and without dimensionality reduction on POS, NER, and Chunking tasks in the CoNLL 2003 data set. It also shows the impact of n-gram concatenation, and dimensionality reduction . Results are reported for different labeled data set sizes, which demonstrate a low-resourced setting.

Since there were no pre-trained embeddings available for WANG2VEC, we trained from the first billion characters from Wikipedia for English. This lead to an sub optimal results across all tasks, hence we have omitted from reporting it.

As indicated by the results in Table 1, it is evident that ELMO performs much better than FASTTEXT for all the tasks and all the data set sizes. While n-gram concatenation or dimensionality reduction did not show compelling results when used in isolation, when combined they contributed to a significant performance gain for both FASTTEXT and ELMO.

In this experiment, we used Annoy approximate nearest neighbor algorithm to quickly calculate the nearest neighbors. Benchmarks done on ANN (Aumüller et al., 2017) have shown accuracy drops when the dimension increases above 100. This can be seen in our results - with concatenated vectors or high dimension vectors like ELMO the accuracy is considerably lower. Since our approach was transductive, we were wary of the efficiency and timing. Traditional k-NN algorithms gave better scores but lead to high time and memory consumption.

Tables 2 and 3 show the results of similar experiments carried out for Sinhala and Tamil POS tagging tasks, respectively. While FASTTEXT performs better than WANG2VEC for Tamil, the opposite was noted for Sinhala. We attribute this difference to the differences in the models created for the two languages - WANG2VEC and FAST-

	POS					Chunking					NER				
	20K	40K	60K	80K	100K	20K	40K	60K	80K	100K	20K	40K	60K	80K	100K
FASTTEXT															
A	0.75	0.79	0.83	0.839	0.81	0.66	0.70	0.73	0.73	0.71	0.35	0.30	0.46	0.46	0.34
B	0.69	0.72	0.74	0.77	0.74	0.66	0.69	0.67	0.72	0.74	0.31	0.25	0.44	0.43	0.35
C	0.60	0.64	0.68	0.70	0.66	0.53	0.57	0.57	0.69	0.60	0.38	0.30	0.44	0.46	0.39
D	0.85	0.88	0.87	0.88	0.86	0.79	0.83	0.85	0.83	0.83	0.61	0.53	0.69	0.66	0.50
ELMO															
A	0.84	0.84	0.88	0.88	0.86	0.82	0.85	0.85	0.82	0.84	0.70	0.67	0.84	0.81	0.65
B	0.90	0.91	0.92	0.92	0.91	0.82	0.83	0.84	0.83	0.84	0.69	0.65	0.76	0.79	0.70
C	0.74	0.76	0.83	0.81	0.77	0.76	0.80	0.79	0.78	0.78	0.62	0.56	0.81	0.77	0.57
D	0.928	0.934	0.941	0.942	0.93	0.90	0.91	0.92	0.88	0.90	0.79	0.76	0.86	0.89	0.70

Table 1: Comparison of different methods to represent nodes and their respective accuracy for different tasks in English. A - Single Vector, B - Dimension reduced Single Vector, C - Concatenated n-gram vectors, D - Dimension reduced concatenated n-gram vectors.

	Tamil POS			Sinhala POS				
	20K	40K	60K	20K	40K	60K	80K	100K
FASTTEXT								
A	0.77	0.81	0.73	0.80	0.76	0.83	0.82	0.77
B	0.62	0.79	0.77	0.80	0.77	0.83	0.82	0.79
C	0.54	0.58	0.58	0.66	0.60	0.67	0.66	0.59
D	0.87	0.88	0.89	0.901	0.88	0.88	0.85	0.84
WANG2VEC								
A	0.72	0.74	0.70	0.815	0.775	0.84	0.81	0.77
B	0.59	0.71	0.54	0.78	0.76	0.81	0.79	0.77
C	0.58	0.82	0.57	0.714	0.66	0.70	0.70	0.63
D	0.70	0.71	0.72	0.801	0.76	0.84	0.85	0.81

Table 2: Comparison of different methods to represent nodes and their respective accuracy for Tamil and Sinhala POS tagging. A - Single Vector, B - Dimension reduced Single Vector, C - Concatenated n-gram vectors, D - Dimension reduced concatenated n-gram vectors.

TEXT models for Sinhala were created using a much larger corpus than that for Tamil. Moreover, domain-similarity was much higher between the Sinhala test data and the data used to build the models. In line with the observation for English, for both the languages, FastText performs better when concatenated and dimensionality is reduced. However, contrary to our expectations, the same is not clearly observed with respect to WANG2VEC.

We then compared the performance of our GSSL approach against Peters et al. (2017) using the best result reported in Table 1. As shown in Figures 1, 2 and 3, when the ELMo model with n-gram concatenation and dimensionality reduction is used, our GSSL approach outperforms Peters et al. (2017)'s bidirectional LSTM CRF.

According to these Figures, when increasing training data, opposed to our expectations there are some drops in scores. One of the glaring one was with NER. For dimension reduced concatenated ELMO vector with 80K training data re-

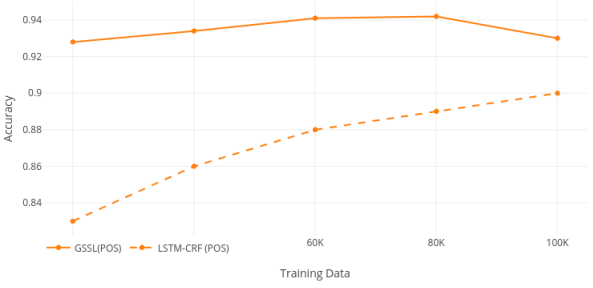

Figure 1: POS accuracy for GSSL Vs LSTM-CRF

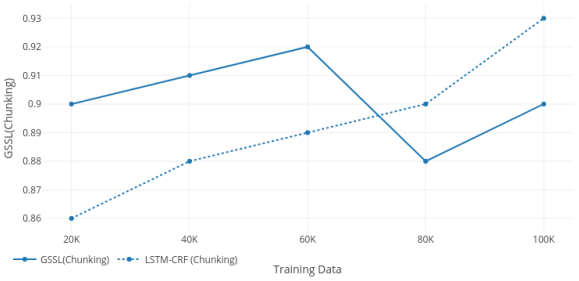

Figure 2: Chunking F1-Score for GSSL Vs LSTM-CRF

Figure 3: NER F1-Score for GSSL Vs LSTM-CRF

sulted an 0.89 F1 score, and it drops to 0.7 for 100K training data. Further analysis revealed that when training data set was increased, it had lead to over-fitting. For example our training data had *Germany* as *LOC* and the test data had *German* which was supposed to be classified as *MISC* was classified as *LOC* due to the close proximity of vectors.

Fernando et al. (2016) had presented POS tagger for Sinhala using hand crafted language dependent features. This research reported the best accuracy for the University of Moratuw corpus. We sampled out a 20K dataset form this corpus as training data for both ours and Fernando et al. (2016)'s approach. The SVM Tagger reported an accuray of 87.11% while we were able to achieve an accuracy of 90.1%. Mokanarangan et al. (2018) had reported for GSSL based approach for FIRE POS tagging with an accuracy of 87.43% for 60K data. For the same training data we were able to achieve an accuracy of 89%.

6 Conclusion

The aim of this research was to develop an efficient GSSL solution for sequence tagging. Our solution is based on identifying neural word embedding models that better capture local context information in graph vertices, and producing a graph in a low-dimensional space that has vertices belonging to different classes well-separated. While some of the word embedding models employed did not generate the expected result, in general, our hypothesis of capturing context information by concatenating vectors is validated. In particular, n-gram concatenation and dimensionality reduction resulted in significant performance gains. Given the fact that our best result outperforms the existing state-of-the-art (for high resource settings), when the labeled data set size is small, our GSSL solution can be presented as a promising alternative for sequence tagging in low-resourced languages.

In the current implementation, LDA calculations are done mostly in memory. Thus when we attempt to use larger annotated training sets with each vector having over 900 dimensions leads to memory overflows. Since our target was towards addressing low resource settings, we did not attempt to address this issue. Thus scalability of our approach for high resource settings should be explored with more optimal dimensionality reduction approaches.

References

Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2017. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In *International Conference on Similarity Search and Applications*, pages 34–49. Springer.

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

E.: Annoy Bernhardsson. 2018. Annoy - Approximate Nearest Neighbor. `https://github.com/spotify/annoy`. [Online; accessed 21-Feb-2018].

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *CoRR*, abs/1606.04596.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 600–609. Association for Computational Linguistics.

Saner Demirel. 2017. *Spectral Graph Convolutional Networks for Part-of-Speech Tagging*. Ph.D. thesis, Universität Koblenz-Landau.

Paramveer S Dhillon, Partha Pratim Talukdar, and Koby Crammer. 2010. Inference driven metric learning (idml) for graph construction.

Sandareka Fernando, Surangika Ranathunga, Sanath Jayasena, and Gihan Dias. 2016. Comprehensive part-of-speech tag set and svm based pos tagger for sinhala. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 173–182.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 583–592. Association for Computational Linguistics.

Luheng He, Jennifer Gillenwater, and Ben Taskar. 2013. Graph-based posterior regularization for semi-supervised structured prediction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 38–46.

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

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Prasenjit Majumder, Mandar Mitra, Dipasree Pal, Ayan Bandyopadhyay, Samaresh Maiti, Sukanya Mitra, Aparajita Sen, and Sukomal Pal. 2008. Text collections for fire. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 699–700. ACM.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Rada F. Mihalcea and Dragomir R. Radev. 2011. *Graph-based Natural Language Processing and Information Retrieval*, 1st edition. Cambridge University Press, New York, NY, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Thayaparan Mokanarangan, Ranathunga Surangika, and Thayasivam Uthayasanker. 2018. Graph based semi-supervised learning approach for tamil pos tagging. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *CoRR*, abs/1705.00108.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: New approach to ad hoc ir. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 59–68, New York, NY, USA. ACM.

François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1702–1712.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050.

Konstantinos Skianis, François Rousseau, and Michalis Vazirgiannis. 2016. Regularizing text categorization with clusters of words. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1827–1837.

Amarnag Subramanya and Jeff Bilmes. 2008. Soft-supervised learning for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1090–1099, Stroudsburg, PA, USA. Association for Computational Linguistics.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 167–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Masashi Sugiyama. 2006. Local fisher discriminant analysis for supervised dimensionality reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 905–912, New York, NY, USA. ACM.

Partha Pratim Talukdar and Fernando Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1473–1481, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, ConLL '00, pages 127–132, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 321–328, Cambridge, MA, USA. MIT Press.

Xiaojin Zhu. 2005. Semi-supervised learning literature survey.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 912–919. AAAI Press.

Bibliography

[1] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[2] Beth M Sundheim. Overview of results of the muc-6 evaluation. In *Proceedings of the 6th conference on Message understanding*, pages 13–31. Association for Computational Linguistics, 1995.

[3] Diego Mollá, Menno Van Zaanen, Steve Cassidy, et al. Named entity recognition in question answering of speech data. 2007.

[4] Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, and Myung-Gil Jang. Fine-grained named entity recognition using conditional random fields for question answering. In *Asia Information Retrieval Symposium*, pages 581–587. Springer, 2006.

[5] Einat Minkov, Richard C Wang, and William W Cohen. Extracting personal names from email: Applying named entity recognition to informal text. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 443–450. Association for Computational Linguistics, 2005.

[6] Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. Twiner: named entity recognition in targeted twitter stream. In *Proceedings of the 35th international ACM SIGIR conference on*

Research and development in information retrieval, pages 721–730. ACM, 2012.

[7] Giridhar Kumaran and James Allan. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304. ACM, 2004.

[8] Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. Named entity recognition approaches. *International Journal of Computer Science and Network Security*, 8(2):339–344, 2008.

[9] James R Curran and Stephen Clark. Language independent ner using a maximum entropy tagger. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 164–167. Association for Computational Linguistics, 2003.

[10] Rahul Sharnagat. Named entity recognition: A literature survey. *Center For Indian Language Technology*, 2014.

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[12] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *CoRR*, abs/1705.00108, 2017. URL `http://arxiv.org/abs/1705.00108`.

[13] Pranavan Theivendiram, Megala Uthayakumar, Nilusija Nadarasamoorthy, Mokanarangan Thayaparan, Sanath Jayasena, Gihan Dias, and Surangika Ranathunga. Named-entity-recognition (ner) for tamil language using margin-infused relaxed algorithm (mira). In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 465–476. Springer, 2016.

[14] SAPM Manamini, AF Ahamed, RAEC Rajapakshe, GHA Reemal, S Jayasena, GV Dias, and S Ranathunga. Ananya-a named-entity-recognition (ner) system for sinhala language. In *Moratuwa Engineering Research Conference (MERCon), 2016*, pages 30–35. IEEE, 2016.

[15] JK Dahanayaka and AR Weerasinghe. Named entity recognition for sinhala language. In *Advances in ICT for Emerging Regions (ICTer), 2014 International Conference on*, pages 215–220. IEEE, 2014.

[16] R Vijayakrishna and L Sobha. Domain focused named entity recognizer for tamil using conditional random fields. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*, 2008.

[17] Dan Garrette, Jason Mielens, and Jason Baldridge. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 583–592. Association for Computational Linguistics, 2013. URL `http://www.aclweb.org/anthology/P13-1057`.

[18] Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Semi-supervised learning for neural machine translation. *CoRR*, abs/1606.04596, 2016. URL `http://arxiv.org/abs/1606.04596`.

[19] Amarnag Subramanya and Jeff Bilmes. Soft-supervised learning for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1090–1099, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1613715.1613857`.

[20] Mouiad Fadiel Alawneh and Tengku Mohd Sembok. Rule-based and example-based machine translation from english to arabic. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on*, pages 343–347. IEEE, 2011.

[21] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16 (2):79–85, 1990.

[22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

[23] Pasindu Tennage, Prabath Sandaruwan, Malith Thilakarathne, Achini Herath, and Surangika Ranathunga. Handling Rare Word Problem using Synthetic Training Data for Sinhala and Tamil Neural Machine Translation. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

[24] Fathima Farhath, Pranavan Theivendiram, Surangika Ranathunga, Sanath Jayasena, and Gihan Dias. Improving domain-specific SMT for low-resourced languages using data from different domains. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

[25] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007.

[26] Behrang Mohit. Named entity recognition. In *Natural language processing of semitic languages*, pages 221–245. Springer, 2014.

[27] Hristo Tanev, Jakub Piskorski, and Martin Atkinson. Real-time news event extraction for global crisis monitoring. In *International Conference on Application of Natural Language to Information Systems*, pages 207–218. Springer, 2008.

[28] Sriparna Saha, Asif Ekbal, and Utpal Kumar Sikdar. Named entity recognition and classification in biomedical text using classifier ensemble. *International journal of data mining and bioinformatics*, 11(4):365–391, 2015.

[29] Prasenjit Majumder, Mandar Mitra, Dipasree Pal, Ayan Bandyopadhyay, Samaresh Maiti, Sukanya Mitra, Aparajita Sen, and Sukomal Pal. Text collections for fire. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 699–700. ACM, 2008.

[30] Douglas E Appelt, Jerry R Hobbs, John Bear, David Israel, Megumi Kameyama, David Martin, Karen Myers, and Mabry Tyson. Sri international fastus system: Muc-6 test results and analysis. In *Proceedings of the 6th conference on Message understanding*, pages 237–248. Association for Computational Linguistics, 1995.

[31] Ralph Grishman. Information extraction: Techniques and challenges. In *International Summer School on Information Extraction*, pages 10–27. Springer, 1997.

[32] Michal Konkol and Miloslav Konopík. Named entity recognition for highly inflectional languages: effects of various lemmatization and stemming approaches. In *International Conference on Text, Speech, and Dialogue*, pages 267–274. Springer, 2014.

[33] Daniel M Bikel, Richard Schwartz, and Ralph M Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1-3):211–231, 1999.

[34] G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3): 268–278, 1973.

[35] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for*

Computational Linguistics, pages 473–480. Association for Computational Linguistics, 2002.

[36] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[37] Label bias problem. `https://cs.nyu.edu/courses/spring13/CSCI-GA.2590-001/LabelBias.pptx`. Accessed: 2018-06-17.

[38] Paul McNamee and James Mayfield. Entity extraction without language-specific resources. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics, 2002.

[39] Sriparna Saha and Asif Ekbal. Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition. *Data & Knowledge Engineering*, 85:15–39, 2013.

[40] Amarnag Subramanya, Slav Petrov, and Fernando Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 167–176, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1870658.1870675`.

[41] Partha Pratim Talukdar and Fernando Pereira. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1473–1481, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1858681.1858830`.

[42] Xavier Carreras, Lluís Màrquez, and Lluís Padró. A simple named entity extractor using adaboost. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 152–155. Association for Computational Linguistics, 2003.

[43] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[44] Amarnag Subramanya and Partha Pratim Talukdar. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014.

[45] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[46] Enrique Alfonseca and Suresh Manandhar. An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 1st international conference on general WordNet, Mysore, India*, pages 34–43, 2002.

[47] Yusuke Shinyama and Satoshi Sekine. Named entity discovery using comparable news articles. In *Proceedings of the 20th international conference on Computational Linguistics*, page 848. Association for Computational Linguistics, 2004.

[48] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

[49] Ryan Cotterell and Kevin Duh. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 91–96, 2017.

[50] Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1946–1958, 2017.

[51] Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, 2016.

[52] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[53] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[54] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[55] Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*, 2017.

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

[57] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[58] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE, 2011.

[59] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[60] CS Malarkodi, RK Pattabhi, and Lalitha Devi Sobha. Tamil ner–coping with real time challenges. In *24th International Conference on Computational Linguistics*, page 23, 2012.

[61] Wei Li and Andrew McCallum. Rapid development of hindi named entity recognition using conditional random fields and feature induction. *ACM*

Transactions on Asian Language Information Processing (TALIP), 2(3): 290–294, 2003.

[62] Sujan Kumar Saha, Sudeshna Sarkar, and Pabitra Mitra. A hybrid feature set based maximum entropy hindi named entity recognition. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, 2008.

[63] Anup Patel, Ganesh Ramakrishnan, and Pushpak Bhattacharya. Relational learning assisted construction of rule base for indian language ner. *Proceedings of ICON*, 2009:7th, 2009.

[64] Karthik Gali, Harshit Surana, Ashwini Vaidya, Praneeth Shishtla, and Dipti Misra Sharma. Aggregating machine learning and rule based heuristics for named entity recognition. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*, 2008.

[65] Animesh Nayan, B Ravi Kiran Rao, Pawandeep Singh, Sudip Sanyal, and Ratna Sanyal. Named entity recognition for indian languages. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*, 2008.

[66] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[67] Hamish Cunningham, Valentin Tablan, Angus Roberts, and Kalina Bontcheva. Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLoS computational biology*, 9(2): e1002854, 2013.

[68] Steven Bird and Edward Loper. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics, 2004.

[69] Ralph Grishman and Beth Sundheim. Design of the muc-6 evaluation. In *Proceedings of the 6th conference on Message understanding*, pages 1–11. Association for Computational Linguistics, 1995.

[70] Rada F. Mihalcea and Dragomir R. Radev. *Graph-based Natural Language Processing and Information Retrieval*. Cambridge University Press, New York, NY, USA, 1st edition, 2011. ISBN 0521896134, 9780521896139.

[71] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of computer science, 2005.

[72] Hany Hassan and Arul Menezes. Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1577–1586, 2013.

[73] Linhong Zhu, Sheng Gao, Sinno Jialin Pan, Haizhou Li, Dingxiong Deng, and Cyrus Shahabi. Graph-based informative-sentence selection for opinion summarization. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 408–412, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2240-9. doi: 10.1145/2492517.2492651. URL `http://doi.acm.org/10.1145/2492517.2492651`.

[74] Micha Elsner and Eugene Charniak. Disentangling chat. *Comput. Linguist.*, 36(3):389–409, September 2010. ISSN 0891-2017. doi: 10.1162/coli_a_00003. URL `http://dx.doi.org/10.1162/coli_a_00003`.

[75] Dipanjan Das and Slav Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 600–609. Association for Computational Linguistics, 2011.

[76] Luheng He, Jennifer Gillenwater, and Ben Taskar. Graph-based posterior regularization for semi-supervised structured prediction. In *Proceedings of*

the Seventeenth Conference on Computational Natural Language Learning, pages 38–46, 2013.

[77] Saner Demirel. *Spectral Graph Convolutional Networks for Part-of-Speech Tagging*. PhD thesis, Universität Koblenz-Landau, 2017.

[78] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 238–247, 2014.

[79] Stefan Evert. The statistics of word cooccurrences: word pairs and collocations. 2005.

[80] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 16 January 2013.

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

[82] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[83] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[84] Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015.

[85] Language models, word2vec, and efficient softmax approximations. `http://rohanvarma.me/Word2Vec/`. Accessed: 2018-06-17.

[86] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

[87] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

[88] Chris Callison-Burch, David Talbot, and Miles Osborne. Statistical machine translation with word-and sentence-aligned parallel corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 175. Association for Computational Linguistics, 2004.

[89] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

[90] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

[91] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[92] Krzysztof Wołk and Danijel Koržinek. Comparison and adaptation of automatic evaluation metrics for quality assessment of re-speaking. *arXiv preprint arXiv:1601.02789*, 2016.

[93] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc., 2002.

[94] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

[95] Ruvan Weerasinghe. A statistical machine translation approach to sinhala-tamil language translation. *Towards an ICT enabled Society*, page 136, 2003.

[96] Roni Rosenfeld and Philip Clarkson. Statistical language modeling using the cmu-cambridge toolkit. 1997.

[97] Sakthithasan Sripirakas, AR Weerasinghe, and Dulip L Herath. Statistical machine translation of systems for sinhala-tamil. In *Advances in ICT for Emerging Regions (ICTer), 2010 International Conference on*, pages 62–68. IEEE, 2010.

[98] Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

[99] Language technology research lab - university of colombo school of computing. `http://ucsc.cmb.ac.lk/ltrl/projects/`. Accessed: 2018-06-17.

[100] Randil Pushpananda, Ruvan Weerasinghe, and Mahesan Niranjan. Sinhala-tamil machine translation: Towards better translation quality. In *Proceedings of the Australasian Language Technology Association Workshop 2014*, pages 129–133, 2014.

[101] S Rajpirathap, S Sheeyam, K Umasuthan, and Amalraj Chelvarajah. Real-time direct translation system for sinhala and tamil languages. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, pages 1437–1443. IEEE, 2015.

[102] Randil Pushpananda, Ruvan Weerasinghe, and Mahesan Niranjan. Statistical machine translation from and into morphologically rich and low

resourced languages. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 545–556. Springer, 2015.

[103] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3, 2007.

[104] Fei Huang and Stephan Vogel. Improved named entity translation and bilingual named entity extraction. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pages 253–258. IEEE, 2002.

[105] Reinhard Rapp. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 519–526. Association for Computational Linguistics, 1999.

[106] Stephen Wan and Cornelia Maria Verspoor. Automatic english-chinese name transliteration for development of multilingual resources. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 1352–1356. Association for Computational Linguistics, 1998.

[107] GAO Wei. Phoneme based statistical transliteration of foreign names for oov problem. *Master's Thesis, The Chinese University of Hong Kong*, 2004.

[108] Paola Virga and Sanjeev Khudanpur. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition-Volume 15*, pages 57–64. Association for Computational Linguistics, 2003.

[109] Long Jiang, Ming Zhou, Lee-Feng Chien, and Cheng Niu. Named entity translation with web mining and transliteration. In *IJCAI*, volume 7, pages 1629–1634, 2007.

[110] Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. Named entity transliteration. *International Journal of Computer Processing of Oriental Languages*, 20(04):289–310, 2007.

[111] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[112] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[113] Paramveer S Dhillon, Partha Pratim Talukdar, and Koby Crammer. Inference driven metric learning (idml) for graph construction. 2010.

[114] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 912–919. AAAI Press, 2003. ISBN 1-57735-189-4. URL `http://dl.acm.org/citation.cfm?id=3041838.3041953`.

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

[116] Open tamil. `https://github.com/Ezhil-Language-Foundation/open-tamil`. Accessed: 2018-06-17.

[117] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[118] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[119] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[120] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[121] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[122] Tomáš Mikolov. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 2012.

[123] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[124] Radu Soricut and Franz Och. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, 2015.

[125] Juan Miguel Cejuela, Peter McQuilton, Laura Ponting, Steven J Marygold, Raymund Stefancsik, Gillian H Millburn, and Burkhard Rost. tagtog: interactive and text-mining-assisted annotation of gene mentions in plos full-text articles. *Database*, 2014, 2014.

[126] Allennlp. `https://github.com/allenai/allennlp`. Accessed: 2018-06-17.

[127] Uom allen nlp repo. `https://github.com/Mokanarangan/UOM-Alle`. Accessed: 2018-06-28.

[128] Tensorflow implementation of contextualized word representations from bidirectional language models. `https://github.com/allenai/bilm-tf`. Accessed: 2018-06-17.

[129] Language models. `https://web.stanford.edu/class/cs124/lec/languagemodeling.pdf`. Accessed: 2018-06-17.

[130] Pre-trained word vectors. `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`. Accessed: 2018-06-17.

[131] Wang2vec. `https://github.com/wlin12/wang2vec`. Accessed: 2018-06-17.

[132] metric-learn: Metric learning in python. `http://metric-learn.github.io/metric-learn/`. Accessed: 2018-06-17.

[133] E.: Annoy Bernhardsson. Annoy - Approximate Nearest Neighbor. `https://github.com/spotify/annoy`, 2018. [Online; accessed 21-Feb-2018].

[134] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In *International Conference on Similarity Search and Applications*, pages 34–49. Springer, 2017.

[135] Masashi Sugiyama. Local fisher discriminant analysis for supervised dimensionality reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 905–912, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143958. URL `http://doi.acm.org/10.1145/1143844.1143958`.

[136] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 321–328, Cambridge, MA, USA, 2003. MIT Press. URL `http://dl.acm.org/citation.cfm?id=2981345.2981386`.

[137] Moses. `http://www.statmt.org/moses/?n=Advanced.Hybrid`. Accessed: 2018-06-17.

[138] Sandareka Fernando, Surangika Ranathunga, Sanath Jayasena, and Gihan Dias. Comprehensive part-of-speech tag set and svm based pos tagger for sinhala. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 173–182, 2016.

[139] Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics, 2011.

[140] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics, 2003.

[141] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.