Shahmukhi Named Entity Recognition by using Contextualized Word Embeddings

Amina Tehseen[a,*], Toqeer Ehsan[b,*], Hannan Bin Liaqat[c], Xiangjie Kong[d,**], Amjad Ali[e], Ala Al-Fuqaha[e,**]

[a]*Department of Information Technology, University of Gujrat, Gujrat, 50700, Pakistan Pakistan*
[b]*Department of Computer Science, University of Gujrat, Gujrat, 50700, Pakistan Pakistan*
[c]*Department of Information Technology, Division of Science & Technology, University of Education, Township Campus, Lahore, 54000, Pakistan*
[d]*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, 310023, China*
[e]*Information and Computing Technology (ICT) Division, College of Science and Engineering (CSE), Hamad Bin Khalifa University, Doha, Qatar*

Abstract

Named Entity Recognition (NER) is an imperative Natural Language Processing (NLP) task which intents to identify and classify predefined named entities in a given span of text. For many Western and Asian languages, NER is a systematically premeditated and established task, however, a little work has been done for Shahmukhi. This paper presents Shahmukhi NER with four key contributions. First, a Bi-directional Long-Short Term Memory (BiLSTM) network based NER model has been developed by incorporating various features including character and word embeddings and Part of Speech (POS) tagging. Second, transfer learning has been employed by training context-free Word2Vec and contextualized Embeddings from Language Models (ELMo) word representations. The word representations have been trained using a Shahmukhi corpus of 14.9 million words. Third, we prepared a cleaner version of an existing Shahmukhi NER corpus by performing Unicode normalization and tokenization errands.

*These authors contributed equally to this work.
**Corresponding author
Email addresses: aminatehseen004@gmail.com (Amina Tehseen), toqeer.ehsan@uog.edu.pk (Toqeer Ehsan), hannanliaqat@hotmail.com (Hannan Bin Liaqat), xjkong@ieee.org (Xiangjie Kong), amsali@hbku.edu.qa (Amjad Ali), aalfuqaha@hbku.edu.qa (Ala Al-Fuqaha)

The corpus has been deduplicated and results are reported on an unseen evaluation set which produced valid results. Fourth, we have studied the impact of two annotation schemes; Inside-Outside (IO) and Inside-Outside-Beginning (IOB) for Shahmukhi. Transfer learning was quite helpful to enhance the performance of NER models especially ELMo embeddings significantly improved the results by prompting contextualized embedding vectors. This is the first study to use character embeddings, POS tagging and transfer learning for Shahmukhi named entity recognition. The IO scheme based model achieved an accuracy of 98.60% with an f-score of 83.75. The IOB scheme based model performed with an accuracy of 98.43% and an f-score of 75.55. These scores are quite promising for an under-resourced morphologically-rich language.

Keywords: Shahmukhi, Punjabi, Named Entity Recognition, Neural Networks, ELMo

1. Introduction

Named Entity Recognition (NER) is a Natural Language Processing (NLP) task to ascertain and excerpt spans of text allied with Named Entities (NEs). NEs are further classified in predefined semantic categories (i.e., name, person, location etc). Multiple annotation schemes are used for NER tagging, (i.e., Inside-Outside (IO), Inside-Outside-Beginning (IOB), Inside-Outside-End (IOE), Beginning-Inside (BI), Beginning-Inside-Ending-Single (BIES) etc.) and each annotation scheme impacts the result of automatic NER. However, NER is considered an indispensable preliminary step for developing various NLP applications, such as Information Retrieval (IR), question answering system (Mollá et al., 2006; Cowan et al., 2015), automatic text summarization (Khademi & Fakhredanesh, 2020), information extraction (Popovski et al., 2020), and machine translation (Vu et al., 2020) etc. Punjabi is a low-resourced South Asian language with 150 million speakers all around the world. Being the 10^{th} most spoken language in the world, its speakers can be found abundantly in Pakistan, India, USA, England, and Canada (Ahmad et al., 2020; Simons & Fennig, 2017).

It shares a large vocabulary with Urdu, Hindi, and Sanskrit which makes it mutually intelligible (Humayoun & Ranta, 2010; Sharma, 2016). Punjabi can be inscribed in two mutually dissonant scripts; i) Shahmukhi and ii) Gurmukhi. Shahmukhi is a Perso-Arabic script which is written right-to-left and is mostly used in Pakistan (PBS, 2017). Whereas, Gurmukhi is a syllabic script which is written left-to-right and generally used in India. Punjabi users have verbal enunciation of both scripts but they find it challenging when it comes to writing. Shahmukhi has 94.4 million users which makes it utmost used script than Gurmukhi (Ahmad et al., 2020). For Shahmukhi, NER is quite challenging task as it has fewer supporting resources, such as gazetteers, annotated corpora, and specialized dictionaries which are helpful for computational linguistic analysis. Morphological richness of Shahmukhi makes it further exigent as a single lexical item can take various forms. Scarcity of word embeddings and flexible word order further adds to the challenges of the NER task.

Ahmad et al. (2020) worked on Shahmukhi named entity recognition and classification. They developed Shahmukhi NER corpus of 318,275 tokens. For corpus annotation, they selected three named entities; organization (ORG), person (PER) and location (LOC) though remaining tokens were tagged as others (O). The corpus had 16,300 named entities including 11,147 PERs, 2,013 ORGs and 3,140 LOCs. For automatic tagging, they used IO tagging scheme. To demonstrate usability of their developed Shahmukhi NER corpus, they used five supervised learning techniques. Supervised learning techniques had three machine learning techniques including Maximum Entropy (Max-Ent), Hidden Markov Model (HMM) and Conditional Random Fields (CRF) and two deep learning techniques; Neural Networks (NN) and Recurrent Neural Networks (RNN). For RNN and NN, Word2Vec (Mikolov et al., 2013a) model was used which was generated using raw Shahmukhi corpus of 318,275 words. For training-testing sampling, ten-fold cross validation method was used. In their experiments, RNN based model outperformed the traditional machine learning approaches and they stated an f-score of 85.2.

We have identified four limitations in the existing work (Ahmad et al., 2020)

which are; 1) The annotated corpus does not have cleaned text. Several punctuation marks and symbols are attached with words making them difficult to tokenize and tag. 2) All the experiments have been done without Part of Speech (POS) tagging. Ahmad et al. (2020) reported the unavailability of a Shahmukhi POS tagger. 3) The corpus has been annotated by using IOB tagging scheme but the NER experiments have been carried by using only IO scheme. IO is a simplified form of IOB tagging which produces higher NER results. However, IO scheme has a drawback that it can not identify same entities appearing consecutively. 4) For the neural NER experiments, Word2Vec embeddings have been trained on the whole corpus including the evaluation sets which makes the results biased. In this work, we have addressed all these limitations and produced more valid results.

This paper significantly contributes to Shahmukhi NER which is an under-resources South Asian language. The work presents four key contributions which have been given as follows:

- NER experiments have been conducted by developing a Bi-directional Long-Short Term Memory (BiLSTM) network based models using different linguistic features which include character embeddings and POS tagging. A first neural Shahmukhi POS tagger has been used for the tagging of the dataset.

- Transfer learning has been employed by training context-free and contextualized word embeddings on a larger corpus of 14.9 million words. This is first work to perform transfer learning for Shahmukhi NER.

- A more cleaner version of the dataset has been produced by removing word and sentence segmentation errors and special characters. Moreover, Unicode normalization has been performed and valid results are reported on an unseen evaluation set.

- An impact of two well-know annotation schemes; Inside-Outside-Beginning (IOB) and Inside-Outside (IO) have been studied and their results have

been reported.

The code and the final version of the dataset are freely available online[1]. For transliterating Shahmukhi characters into Roman text, we have used a transliteration scheme proposed by Malik et al. (2010). The rest of the paper is organized as follow. Section 2 provides a comprehensive survey of NER for Shahmukhi, Gurmukhi and Urdu with a focus on traditional machine and deep learning research for them. In Section 3, we have discussed Shahmukhi NER challenges. Different NER annotation schemes are presented in Section 4. Section 5 describes the corpus preparation process. In Section 6, the experimental setup is presented. Section 7 discusses the results whereas Section 8 concludes this research by discussing findings.

2. Related Works

In this section, we have presented the work done for Shahmukhi, Gurmukhi and Urdu named entity recognition. We have quoted work from Gurmukhi and Urdu because Gurmukhi is another script of Punjabi whereas Urdu and Shahmukhi use same script for writing.

Named entity recognition task was first time presented at sixth Message Understanding Conference-6 (MUC-6) (Sang & De Meulder, 2003). Extensive NER work has been done for various established languages such as English, Chinese and Japanese etc. Techniques that have been used for NER are rule-based, machine learning, deep learning and hybrid approaches. In rule-based approach, linguists manually write linguistic patterns which distinguish the NEs from text. The grammatical patterns are language dependent in rule-based approaches which make it challenging to explore. For machine learning, a large size of annotated corpus is required for better prediction. Some machine learning techniques that are being used for NER are Hidden Markov Model (HMM),

[1] https://github.com/toqeerehsan/Punjabi-Shahmukhi-Named-Entity-Recognition

Support Vector Machine (SVM), Maximum Entropy (Max-Ent) model, Artificial Neural Networks (ANNs) etc. Deep learning architecture based models perform proficiently as compared to traditional machine learning algorithms.

Shahmukhi is an under-resourced language and minimal work has been done for Shahmukhi NER. Ahmad et al. (2020) published first work on Shahmukhi NER. They developed a corpus of 318,275 tokens with 16,300 named entities including 2,013 ORG, 11,147 PER and 3,140 LOC. They used IO tagging scheme for NE annotation. They used five supervised learning techniques to illustrate usability of their developed NER corpus. Deep learning technique RNN outperformed the traditional machine learning approaches in their experiments. They reported an f-score of 85.2. The reported results are quite high but the developed NER system has limitations. The annotated corpus developed by Ahmad et al. (2020) did not have cleaned text. They performed all the experiments without POS tagging. They performed their experiments using IO annotation scheme only. For the neural NER experiments, Word2Vec embeddings have been trained on the whole corpus including test sets which makes the results biased. Their work and its limitations have been discussed in Section 1 in more details.

Singh (2013) worked on Gurmukhi NER using CRF approach. They used IOB tagging scheme and tagged NER corpus of 17 thousands words. A set of features were used for experiments including context word, word suffix, word prefix, gazetteer list, POS information and NE information. They reported an f-score of 0.8578. Chopra & Morwal (2012) developed Gurmukhi NER system using HMM classifier. They generated corpus from Gurmukhi stories available on the world wide web. Gurmukhi corpus annotation was performed manually with a tag set of nine tags. Their system performed with an f-score of 0.8840.

Kaur & Josan (2014) evaluated Gurmukhi NER system using CRF. They used manually tagged Gurmukhi news corpus, which was tagged with a named entity tag set of 12 tags. IOB annotation scheme was used for tagging. In their experiments, they used feature set comprising context word, POS information, first name, middle name, last name and gazetteers lists. They stated an f-score

of 0.8092.

Kaur & Josan (2015) further performed NER for Gurmukhi by using a dataset of 170,000 words. The test set contained 30,000 words. Their research was based on the idea of evaluating various features which are helpful in NER recognition. The evaluating features were context window of size three, five and seven, infrequent words, length of words and various digits. Experiments were conducted with various combination of features. Their NER system performed best with an f-score of 0.8746 using feature set encompassing word window of size five, length of word feature, infrequent word and digit feature.

Mukund et al. (2010) developed an Information Extraction (IE) system for Urdu. NER task was one of the sub-tasks of their Urdu IE system. They performed experiments using Max-Ent and CRF and reported 55.3 and 68.9 f-score respectively. Singh (2008) developed CRF based NER system for Urdu, Telugu, Bengali, Hindi and Oriya. They reported an f-score of 43.46 for Urdu.

Riaz (2010) developed a rule-based NER system for Urdu. They devised a hand-crafted rule-based NER system due to inadequacy of online resources and unavailability of enough annotated corpora. They chose 2,262 documents from Backer-Riaz corpus. To construct rules, they used 200 documents whereas for evaluation 600 documents were chosen. The proposed NER system, extracted 187 named entities of which 171 were true named entities and reported 91.1 f-score.

Singh et al. (2012) also used rule-based approach for Urdu NER. They used rules to extract date, numbers, time and non-numeral. To classify terms, person names and location they used suffixes matching. They incorporated different gazetteers to get data of locations, names and titles. They used news data for system testing and prepared two test sets. The first test set had 12,032 tokens whereas second test set had 150,243 tokens. The proposed system achieved 60.09% accuracy for first test set and 88.1% accuracy for second test set.

Mukund & Srihari (2009) presented a two-stage and four-stage bootstrapped model for Urdu NER system development. They used CRF technique with 10-fold cross validation. Two-stage model achieved an f-score of 55.3 and four-stage

model attained 68.9 f-score.

Jahangir et al. (2012) developed Urdu NER model using n-gram (unigram and bigram) models. For both n-gram models, gazetteer list was used. The bigram model additionally used smoothing techniques as well. NER system achieved 75.14 f-score with unigram approach and 75.83 f-score using bigram model with Back-off smoothing.

Malik (2017) developed an Urdu NER corpus. They annotated 652,852 tokens which were extracted from 15 different text domains. They used two supervised machine learning algorithms; HMM and ANNs. HMM scored 66.90 f-score whereas ANNs reported 84.17 f-score.

Kanwal et al. (2019) developed Urdu NER corpus of 926,776 tokens which had 99,718 named entities. They generated six word embeddings on two Urdu corpora using three different approaches; fastText, Word2Vec and GloVe (Goldberg & Levy, 2014; Pennington et al., 2014; Bojanowski et al., 2017). Khan et al. (2020) proposed an Urdu NER system using Deep Recurrent Neural Networks (DRNNs) with word embeddings. In their proposed model, they used language dependent features, for instance POS tags and language independent features such as words context window. A package Text2Vec[2] was used to acquire word embeddings. They used seven entity classes; designation, person, date, location, time, person and number. Three datasets were used to illustrate the effectiveness of DRNN models using word embeddings. DRNN approaches achieved 81.1, 79.94 and 63.21 f-score on three benchmark datasets.

The literature presents several different machine and deep learning techniques to perform NER for different languages. To achieve better recognition results, valid annotated datasets are crucial. Deep learning models have shown state-of-the-art results for many NLP tasks but these types of models are data hungry. For low-resourced languages with smaller datasets, transfer learning shows improvements in results (Niu et al., 2020; Liu et al., 2021, 2020). Transfer learning has been deployed in several machine learning domains including

[2]https://github.com/zhong kaifu/Txt2Vec

image processing (Niu et al., 2021; Wang et al., 2021), speech processing (Niu et al., 2020; Sekhar et al., 2022) and natural language processing (Ehsan et al., 2022; Ehsan & Hussain, 2019; Tehseen et al., 2022).

The literature shows that multiple researches have been conducted to perform named entity recognition for Gurmukhi and Urdu. Several datasets are also available for these two languages. However, Shahmukhi is still an under-resourced language and there is a need to develop computational resources including named entity recognition.

3. Shahmukhi Script Properties

In this section, we discuss Shahmukhi script properties which makes NEs identification and classification challenging. For all languages, NE recognition and classification is a challenging task and a number of challenges needs to be addressed. South Asian languages face a few additional NER challenges (Goyal, 2008). Shahmukhi NER challenges include, no capitalization, free word order, spelling variations, agglutinative language, loan words, nested entities, inaccurate tokenization and conjunction ambiguity etc.

3.1. No Capitalization

Shahmukhi and other south Asian languages don't have the concept of capitalization. In European languages such as English, named entities can be identified by capitalization. Lack of capitalization makes Shahmukhi NEs identification strenuous. For instance, due to capitalization rule in English language, acronyms could be identified easily, but it is difficult to identify acronym in Shahmukhi. For example 'BBC' is written in English and *bI bI sI* in Shahmukhi.

3.2. Spelling Variation

Shahmukhi lacks the standardization for spellings. Different writers write according to their own understandings. For instance, the word *Lahore* can

be written as *lOr*, *lAHOr* or *lAhOr* which makes named entities identification challenging.

3.3. Words from Other Languages

Shahmukhi also have many borrowed words from other languages. For example, word *dunyA* 'world' and *misr* 'Egypt' are borrowed from Arabic, *tOlyA* 'towel' and *almArI* 'cupboard' are borrowed from Portuguese and *jindRI* 'life', *acAr* 'pickle' are borrowed from Persian language.

3.4. Free Word Order

Shahmukhi is a free-word order language, in which word order could be changed for a same sentence. For example, the sentence *asI lAhOr gE* 'We went to Lahore' can be written as *lAhOr asI gE*. Therefore, the position of a word may not determine NEs.

3.5. Inaccurate Tokenizer

For every text processing task, accurate tokenization is essential. In western languages, space indicates the words boundary. Detection of word boundary is foremost challenge in Shahmukhi script due to inconsistent use of spaces. Mostly, space is added between two morphemes for readability such as word *mOm battI* 'candle' is composed of two words *mOm* 'wax' and *battI* 'light' as without space, *mOmbatti*, is not likely to readable.

3.6. Agglutinative Language

Agglutinating languages form new words by adding suffixes and prefixes to root words. Shahmukhi being highly agglutinative language can form many words from root word. For instance, *cIn* 'China' to *cInI* 'Chinese'. It is challenging for NER system to classify these words correctly.

4. Annotation Schemes

This section presents a comparison of multiple named entities annotation schemes. Multiple annotation schemes are used for NER tagging such as IO, IOB, IOE, BI, BIES. IOBES etc. Each annotation scheme impacts the result of automatic NER and identification of named entities. In this section, we have tagged sample Shahmukhi sentences with these annotation schemes.

Inside-Outside (IO) is the simplest annotation scheme. One of two tags I and O are assigned to each token in the dataset. The I tag is assigned to named entities whereas O is assigned to other words. For example in (1) *lAhOr* 'Lahore' is the named entity whereas other words such as *jEs* 'that' and *din* 'day' are normal words therefore O tag is assigned to them.

(1) jEs din asI lAhOr gyE
O O O I O
'The day we went to Lahore.'

Inside-Outside-Beginning (IOB) scheme consists of three tags; I, O and B. B tag is assigned to the beginning word of a named entity whereas I indicates inside of a named entity and O is assigned to other words which are not part of any named entity. For instance, in (2) B tag is assigned to *bI* as it is the beginning of the named entity *bI bI sI* 'BBC' whereas I tag is assigned to other subsequent parts of the same named entity. Though, IO is a simplified form of IOB tagging which produces higher NER identification results. However, IO scheme has a drawback that it can not identify same entities appearing consecutively in a text.

(2) aE xbr bI bI sI tE vE
O O B I I O O
'This news is on BBC.'

Inside-Outside-End (IOE) scheme works almost like IOB, but it uses E tag to annotate ending of a named entity instead of the beginning. For example, in (3), E tag is assigned to *sI* C which is end tag for the named entity *bI bI sI* 'BBC'.

(3) aE xbr bI bI sI tE vE
O O I I E O O
'This news is on BBC.'

Inside-Outside-Beginning-End-Single (IOBES) scheme is an alternative of IOB as it enhances amount of information related to named entities boundaries. Additionally, with B (beginning), I (inside), E (ending) and O (other) tags it adds S tag for single token NE. For example, in (4) *lAhOr* 'Lahore' is single token named entity tagged with tag S. Whereas, tags B(beginning), I(inside) and E(end) tag are assigned respectively to a named entity *bI bI sI* 'BBC'.

(4) lAhOr dI xbr bI bI sI tE vE
S O O B I E O O
'Lahore's news is on BBC.'

Beginning-Inside (BI) scheme annotates named entities in similar method like IOB. Moreover, it tags the beginning of a non-entity word with BO tag and remaining as IO. For example in (5), *aE xbr* 'this news' are consecutive non-entity words. The first word *aE* 'this' is tagged as BO as its beginning of non-entity whereas *xbr* 'news' is subsequent non-entity word therefore it is tagged as IO. However, the named entity *bI bI sI* 'BBC' is tagged similar to IOB tagging scheme.

(5) aE xbr bI bI sI tE vE
BO BI B I I BO IO
'This news is on BBC.'

Inside-End (IE) tagging scheme works just like IOE, the difference is that, IE tags the end of a non-entity word with EO tag and the remaining as IO. As in (6), *aE xbr*, 'this news', *aE* 'this' is tagged as IO and EO tag is assigned to *xbr* 'news' which is end word of a non-entity. The named entity *bI bI sI* 'BBC' is tagged similar to IOE tagging scheme as in (3).

(6) aE xbr bI bI si tE vE
IO EO I I E IO EO
'This news is on BBC.'

Beginning-Inside-Ending-Single (BIES) scheme works almost like IOBES. Additionally, it also tags non-entity words using similar pattern. It encodes beginning of non-entity words using BO tag and inside of non-entity words with IO tag. Single non-entity lexical item are given a tag SO. (7) is tagged using BIES tagging scheme.

(7) lAhOr dI xbr bI bI sI tE vE
S BO IO B I E BO IO
'Lahore's news is on BBC.'

The IOB and IO annotation schemes are quite adequate and simple to annotate a corpus for named entities. In both schemes, the words which are not part of any named entity, are tagged with O tag. Other schemes use more number of tags and use complex structures to annotate even other words. IO scheme is simpler as compared to IOB scheme but it can not differentiate two consecutive named entities of same types. However, it helps to produce higher results for automatic NER systems. In this work, we have used well-known IO and IOB annotation schemes. We further studied how these annotation schemes impacts the results.

5. Corpus Preparation

This section presents the corpus preparation steps. For NER task, standardized and cleaned corpora are requisite. Section 5.1 discusses Unicode normalization. Section 5.2 presents text cleaning process in detail. In Section 5.3, POS tagging for the NER corpus is described. Section 5.4 presents the plain Shahmukhi corpus collection and preprocessing. Section 5.5 summarizes the corpus preparation task.

Shahmukhi NER corpus developed by Ahmad et al. (2020) has 318,275 tokens and 16,300 named entities. The corpus is of significant size but after perusal, a few issues were found such as unnormalized Unicodes, incorrect tokenization and sentence segmentation.

5.1. Unicode Normalization

For computational processing of a language, unified Unicodes of corpora are crucial. Shahmukhi alphabets are not standardized, consequently users use different alphabets to write it. Normally, Urdu alphabets are used to write Shahmukhi as both use Perso-Arabic script (Bhurgari, 2007; Hasan et al., 2015). For mapping Shahmukhi Unicodes, we have used a phonetic Urdu keyboard developed by Center for Language Engineering (CLE)[3]. Initially, we developed a utility and generated a report of unnormalized Unicodes. A few mapping examples are shown in Table 1. Essential Unicode mappings were mapped such as '0647' and '06D3' are mapped to '06C1' and '06D2+0654' respectively. A few Sindhi characters which were used in Sindhi words aren't mapped to Urdu characters. Similarly, Hindi characters are also kept intact and are not removed or replaced. However, the diacritics were removed from the corpus.

Table 1: Mapping of characters according to CLE keyboard Unicodes.

From Unicode	To Unicode
0623	0627+0654
0627+0653	0622
0647	06C1
0624	0648+0654
06D3	06D2+0654
003F	061F
060D	060C
2019+2019	0022
2018	0027
002E	002D

[3]http://www.cle.org.pk/software/localization/keyboards/CRULPphonetickb2Lv1.0.htmls

5.2. Text Cleaning

For NER corpus cleaning, we overlooked tokenization of words and sentence segmentation. We particularly checked for correct tokenization of punctuation marks such as comma, colon, semicolon, question mark, sentence end marker, exclamation mark, single quotes, double quotes and brackets. For instance, comma was incorrectly tokenized in three scenarios. It was written before words such as *,faOj* ',military', after words such as *panjAb,* 'Punjab,' and between two words such as *hussEn,lAhOr* 'Hussain,Lahore'. We manually corrected these issues carefully.

A lot of issues were also found for sentence end marker. After tokenization, we performed sentence segmentation as many sentences were not ending on sentence end marker or question mark. We manually checked and corrected them. Similarly non-printable characters were also removed from the corpus. After cleaning, number of tokens in the corpus have been increased.

5.3. Shahmukhi POS Tagging

We further performed POS tagging for the NER corpus by using a neural POS tagger developed by Tehseen et al. (2022). A corpus of 130 thousands words was annotated and a BiLSTM based tagger was trained which achieved an accuracy of 96.12%. The POS tag set has been derived from an existing Urdu tag set (Ahmed et al., 2015) containing 35 tags. The tag set is quite applicable to Shahmukhi as both Urdu and Shahmukhi have similar syntactic categories and script. However, an additional POS tag has been introduced to mark pronominal suffixes of Shahmukhi. The POS tag set of Ahmed et al. (2015) has been used for many NLP tasks like tagging of Urdu Tweets (Baig et al., 2020), phrase chunking (Ehsan et al., 2022), constituency treebanking (Ahmed et al., 2020) and parsing (Ehsan & Hussain, 2020, 2019, 2022), dependency parsing (Ehsan & Butt, 2020), topic modeling (Ehsan & Asif, 2018), break index annotation of a speech corpus (Mumtaz et al., 2016) and sense tagging (Urooj et al., 2014). Our Shahmukhi NER model is trained by using POS tags along with other features.

5.4. Plain Shahmukhi Corpus

A plain Shahmukhi corpus was collected from online sources which include; Wichaar[4], Bhulekha[5], Punjabi Stories[6], Enabling Minority Language Engineering (EMILLE)[7] and Punjabi Wikipedia[8]. After corpus collection, Unicode normalization and word and sentence segmentation have been performed. The corpus contains 14.9 million words from 14 different genres based on the contents. It is the first large digital corpus of Shahmukhi which has been used for transfer learning to develop Shahmukhi NER system.

The detailed statistics of the corpus are presented in Table 2. The textual data from Wichaar, Bhulekha and Punjabi stories have been classified into separate text domains whereas data from EMILLE and Wikipedia are handled as individual domains as both have plain text. Emille dataset was the largest domain with five million tokens whereas science and technology was the smallest domain with 36 thousands tokens.

5.5. Summary of Preprocessing

In Section 3, we have discussed some Shahmukhi script properties which make NEs identification and classification challenging. Shahmukhi NER challenges include; no capitalization, spelling variation, free word order, words from other languages, inaccurate tokenizer and agglutinative language etc. We have carefully catered these issues in our research. For spelling variations, free word order, words from other languages and agglutinative aspect of the language, we have performed transfer learning as it covers the morphological aspect of the language. For transfer learning, the word embeddings have been trained on a large corpus of Shahmukhi. It included a large vocabulary in our models which was helpful to solve the Out-Of-Vocabulary (OOV) problem. The large corpus

[4]http://www.wichaar.com/
[5]https://bhulekhatv.com/
[6]http://www.punjabikahani.punjabi-kavita.com/Punjabi-StoriesShahmukhi.php
[7]https://cqpweb.lancs.ac.uk/emillewpun/
[8]https://pnb.wikipedia.org/wiki/

Table 2: Text domains with total number of tokens and sentences in the plain Shahmukhi corpus.

Sr. #	Domain	# Tokens	# Sentences
1	Artists	80,208	3,857
2	Business and Economy	58,819	2,263
3	Comics	45,713	2,108
4	News	956,403	29,357
5	Sports	45,080	1,627
6	Short Stories	2,527,736	162,214
7	Science and Technology	36,449	1,316
8	Women	47,512	2,076
9	Novels	96,482	7,087
10	Punjabi Language	112,631	4,588
11	Articles	550,541	27,441
12	Classics	338,196	15,217
13	EMILLE Dataset	5,218,157	315,021
14	Pnb Wikipedia	4,883,034	223,762
	Total	14,996,961	797,934

size also solved the spelling variations, free word-order, words from other languages and morphological aspect of the language. The results are evident that the transfer learning by training context-free and contextualized word representations are quite helpful to identify named entities for a low-resourced language. For inaccurate tokenization, text cleaning has been performed as discussed in Section 5.2 Furthermore, we manually reviewed the whole corpus for rectifications.

6. Neural NER Model

This section presents the details of our neural NER model which is based on

LSTM networks with further discussion of the experimental setup. For multi-dimensional sequential data processing, recurrent neural networks perform proficiently (Deng, 2014; Du et al., 2020). In RNN, the last state output is used as an input to present state. LSTM is a variant of RNN that undertakes the issue of RNN's long term dependencies. Figure 1 shows the structure of an LSTM cell. LSTM architecture has various memory blocks which are called cell and a chain edifice which comprises four neural networks. Cells retain the information and gates ensure the memory manipulation. There are three gates; Forget gate, input gate and output gate.

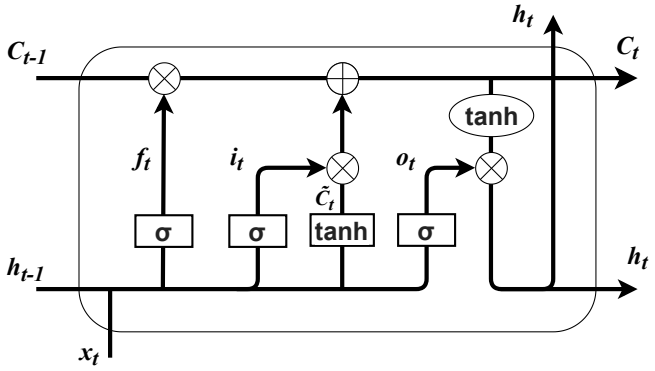

Figure 1: LSTM cell architecture.

The forget gate eradicates the information which is not longer expedient in cell state. Equation 1 is for forget gate f_t which states the information to discard from the cell state. Two inputs, h_{t-1} which is previous cell's output and x_t input at the specific time, are fed to the gate and multiplied with weight matrices w_f and further followed by bias' addition b_f. Then the resultant is passed to *sigmoid* which is an activation function and gives binary output. For a particular cell statue, if it gives zero output then the information is discarded but if output is one then for future usage the information is kept.

$$f_t = \sigma(w_f[h_{(t-1)}, x_t] + b_f) \tag{1}$$

The input gate adds expedient information to the cell state. At first, *sigmoid*

function regulates the information and like forget gate sieves the information to retain using inputs h_{t-1} and x_t. Further, *tanh* function creates a vector that gives output from -1 and +1. To attain useful information, regulated values and vector's values are multiplied. The Equation 2 is for input gate.

$$i_t = \sigma(w_i[h_{(t-1)}, x_t] + b_i) \tag{2}$$

The output gate extracts the useful information from current cell state. At first, *tanh* function is applied on the cell to generate a vector. Then *sigmoid* function regulates the information and sifters the values to retain using input x_t and h_{t-1}. The values of the vector are then multiplied with regulated values and output is generated. The output includes current hidden state h_t and current state c_t which becomes input for the next cell. The Equation 3 represents the output gate.

$$o_t = \sigma(w_o[h_{(t-1)}, x_t] + b_o) \tag{3}$$

The candidate for cell state is calculated by using Equation 4. Between network output and cell state information, w_c is the weight matrix of *tanh* operator. b_c presents the bias vector. The Equation 5 shows cell state at a timestamp *t*. f_t refers to forget gate at timestamp *t*, its value is multiplied with previous timestamp c_{t-1}. i_t presents input gate at timestamp *t*. i_t is multiplied with value generated by *tanh* $c`_t$ The Equation 6 is used for the final output. It calculates current hidden state h_t by using o_t and *tanh* of the updated cell state c^t .

$$c`_t = tanh(w_c[h_{(t-1)}, x_t] + b_c) \tag{4}$$

$$c_t = f_t * c_{t-1} + i_t * c`_t \tag{5}$$

$$h_t = o_t * tanh(c^t) \tag{6}$$

From these equations, it could be inferred that the cell state discerns what it needs to forget from the previous state and what it needs to ruminate from the present timestamp. Some of the prominent applications of LSTM include text generation, image captioning, speech recognition and language translation etc.

Our proposed Shahmukhi NER model is based on bidirectional LSTM networks. BiLSTM networks perform quite promising for sequence labeling as compared to traditional machine learning algorithms. It also accosts the vanishing gradient problem of traditional RNNs (Khan et al., 2019). BiLSTM has two LSTM layers, one reads the sequence in forwards direction whereas the second reads in backward direction. The BiLSTM gets benefit of past and future features as well by attaining additional context to network at specific time frame. The architecture of BiLSTM model that we used is shown in Figure 2. The input sequence of N words x_1,x_2,...,x_n is given as input. A few instances of the vector $h_{1:n}$ are shown in Figure 2 against input vector $x_{1:n}$.

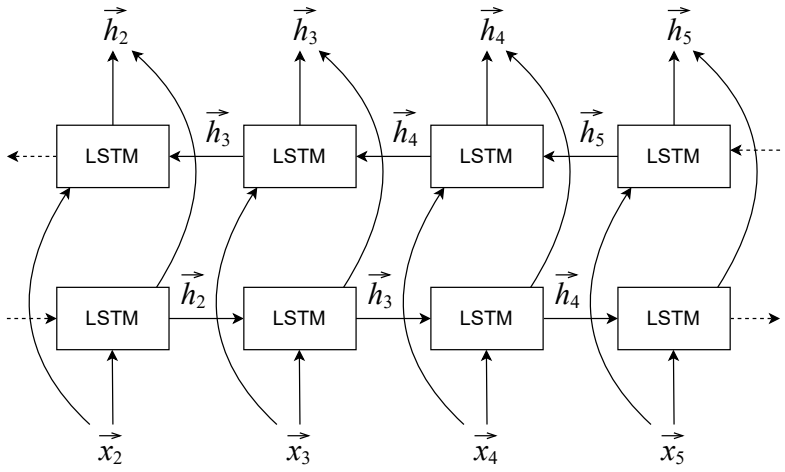

Figure 2: Bi-directional Long-Short Term Memory model for sequence labeling.

The BiLSTM($x_{1:n}$,i) function is defined in Equation 7 where the outputs of backward and forward layers are concatenated. $LSTM_f$ refers to the LSTM layer which reads the sequence label in forward direction whereas $LSTM_r$ reads

in reverse. The function denotes to a vector i by conditioning the past antiquity $x_{1:i}$ and the forthcoming sequence $x_{i:n}$ as well. For each input vector $x_{1:n}$, a few instances of the vector $h_{1:n}$ are presented in Figure 2.

$$BiLSTM(x_{1:n}, i) = LSTM_f(x_{1:i}).LSTM_r(x_{n:i}) \tag{7}$$

At output layer, a *softmax* non-linearity function is employed to perform multi-class classification which returns NEs classes t_1,t_2,...,t_n for each input sequence of N words x_1,x_2,...,x_n as shown in Equation 8.

$$o_i = Softmax(Xh_i + b) \tag{8}$$

Figure 3 shows the neural architecture which has been developed to train the NER model. The dataset contains tokens, POS tags and NE labels. The training samples have been converted to vector representations to input into the neural model. The vector encodings have word embeddings, pre-trained words representations, POS representations and character embeddings. POS feature is pervasively used for NER tasks as they are capable to divulge sentence's grammatical structure (Alshammari & Alanazi, 2020). Character embeddings are helpful to learn the morphology of a language. We have done experiments with different feature sets but Figure 3 shows the generic model for NER. All the vector encodings have been merged into a single vector to input them to a BiLSTM network. Hidden LSTM layers contain 256 units each. We experimented up to three hidden layers. $Softmax$ non-linearity has been used for multi-class classification which produced probabilities for each NE class. Finally, the NE classes are predicted on the basis of maximum likelihood. Predicted NEs have been evaluated against the gold reference corpus which has been annotated manually.

We started our experiments by employing a single BiLSTM layer. Then we extended our model with character embeddings. Later on, we protracted the model with one BiLSTM layer, character embeddings and word embeddings. We also experimented with two and three BiLSTM layers. The model also

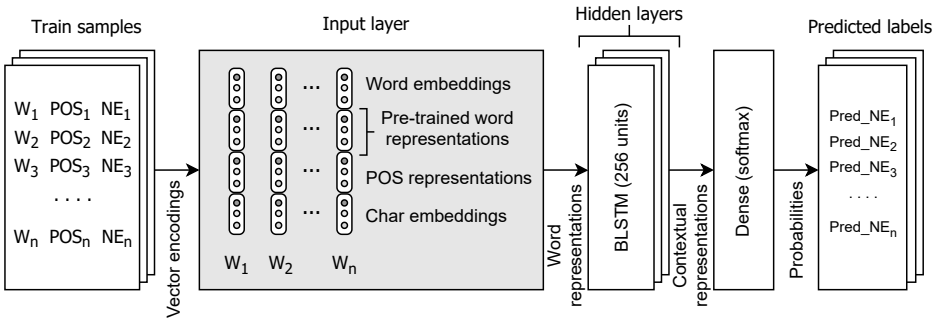

Figure 3: Neural NER model architecture.

has been extended by adding POS tags with one BiLSTM layer and character embeddings. We further extended the model by incorporating character embeddings, Word2Vec and POS with one BiLSTM layer. Later on, two BiLSTM layers were used with the same features.

The BiLSTM model has been further experimented by adding ELMo embeddings. At first, ELMo was integrated with a single BiLSTM layer. Then the model was extended by adding Word2Vec model with ELMo embeddings and experimented with one and two BiLSTM layers. Later on, features were extended and character embeddings were added with Word2Vec and ELMo embeddings and results were reported with one and two BiLSTM layers for these features as well.

The BiLSTM model has been developed by using *keras* library with *TensorFlow* back-end. For all experiments, bidirectional LSTM layers were set to have 256 units. A dropout layer was used after each LSTM layer with a value of 0.2 (20%). Root Mean Squared Propagation (RMSprop) gradient descent optimization algorithm with a learning rate of 0.002 was used along with categorical cross-entropy loss function. The output layer of each experiment performed multi-class classification by employing *softmax* activation function. For character embeddings, maximum character length was set to 20 along with 30 units for character LSTM layer. As BiLSTMs work with fixed length of sentences, therefore, sequence length was set to 181, which was longest sentence

length in the dataset.

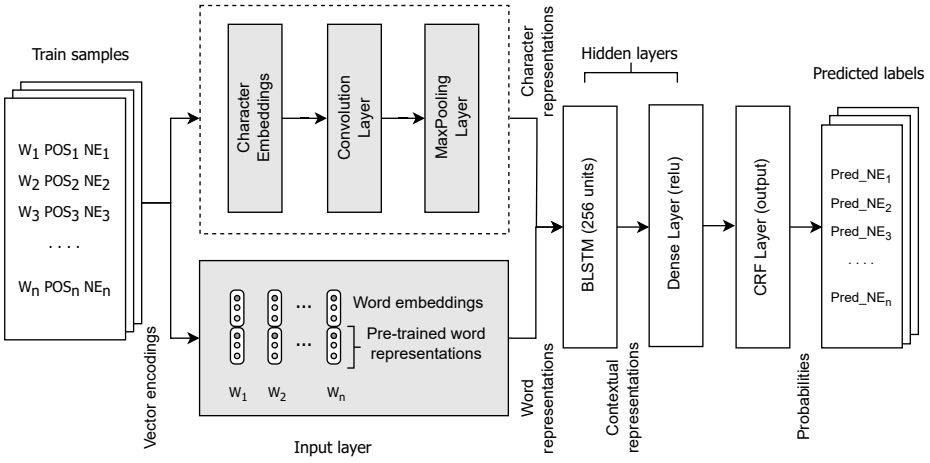

Figure 4: CNN-BiLSTM-CRF model architecture.

The BiLSTM model has been updated to have Convolution Neural Network (CNN) layer along with Conditional Random Field (CRF) output layer. This model produces significant results for sequence labeling tasks and is referred as CNN-LSTM-CRF (Figure 6). The CNN component extracts features from the input sentence by applying filters of different sizes to the character embeddings of a sentence. The output of the CNN is a sequence of feature maps that captures different patterns and structures. The LSTM component is responsible for learning temporal dependencies between words in an input sentence. The output of the LSTM is a sequence of hidden states that represent the context of each word in the sentence. The CRF component takes the sequence of hidden states from the LSTM and applies a linear-chain CRF layer to model the dependencies between the named entities. The output of the CRF is the sequence of named entities in the input sentence. The advantage of using CRF in neural network models is that it allows for modeling of the dependencies between labels, which is important in sequence labeling tasks where neighboring labels can influence each other. Additionally, by optimizing the model using the maximum likelihood estimation of the CRF parameters during training, the model is able

to learn to capture the structure of the data and make accurate predictions. The CNN-LSTM-CRF model for NER is a powerful deep learning architecture that has shown significant performance for NER.

The CNN-LSTM-CRF model has been developed by using *keras* library with *TensorFlow* back-end. In the experiments, the input layer has been divided into word vectors and character embeddings. The character embeddings have been fed to a convolution layer with 32 filters, kernel size of three and *relu* activation. The convolution layer is followed by a MaxPooling layer with pool size of one. On the other hand, the word vectors are given to word embeddings layer with 100 dimensions. The word embedding layer is further enhanced by incorporating pre-trained word representations. Feature vectors from convolution and word embedding layers are fed to a bidirectional LSTM hidden layer which has been set to have 256 units. A dropout layer has been used after the bidirectional LSTM layer with a value of 0.2 (20%). The LSTM layer is followed by a fully connected dense layer with *relu* activation function. And finally, a CRF output layer performed the multi-class classification by using the Root Mean Squared Propagation (RMSprop) gradient descent optimization algorithm with a learning rate of 0.002. For character embeddings, maximum character length was set to 20 along with 50 units for character LSTM layer.

We further integrated transfer learning for Shahmukhi NER tagger. Transfer learning is a technique in which neural network model is trained on a large corpus. It is quite helpful when there is less amount of annotated data especially for a low-resourced language. The trained model aids the neural network to congregate briskly for other tasks (Malte & Ratadiya, 2019). The details of transfer learning have been described in Section 6.1.

6.1. Transfer Learning

Neural models require lot of annotated data to produce state of the art results. Annotation of a huge dataset is quite costly in terms of time and resources. Therefore, transfer learning is a suitable technique to learn word embeddings from a large unannotated corpus which overcomes the problem of

data sparsity. Transfer learning is also helpful to cater the OOV problem. We have trained word embeddings on a plain Shahmukhi corpus containing 14.9 million tokens. To ratify transfer learning in our experiments, we have trained context-free words embedding as well as contextualized word representations which are discussed in Section 6.1.1 and Section 6.1.2 respectively.

6.1.1. Context-Free Word Representations

Context-Free word embeddings produce a single feature vector against each word which means that each word has a single meaning associated with it. However, these words embeddings are capable of learning semantic and syntactic information of a language. Well-known context-free word embeddings are Word2Vec (Mikolov et al., 2013b), fastText (Wang et al., 2018) and GloVe (Pennington et al., 2014). The Word2Vec embeddings can be trained by using two methods which are skip-grams and continuous bag of words (CBoW) model. The CBoW model takes context words as input and predicts the original words within a selected window. The CBOW model maximizes the Equation 9 (Mahdaouy et al., 2016), where w_t is a given target word and [w_{t-c},..., w_{t-1},w_{t+1},..., w_{t+c}] shows its context. |V| refers to the number of words in the corpus and c is the size of dynamic context w_t,

$$\frac{1}{|V|}\sum_{t=1}^{|v|} log[P(w_t|w_{t-c},...,w_{t-1},w_{t+1},...,w_{t+c})] \tag{9}$$

On the other hand, the skip-gram is based on a feed-forward neural network which takes a word as input and learns its context words. To predict the context of the current target word, the skip-gram model uses Equation 10 (Mahdaouy et al., 2016). It maximizes the average log probability for a given sequence of [w_{t-c},..., w_{t+c}] training words. For dynamic context of w_t, c is the size and |V| refers to number of words in the corpus.

$$\frac{1}{|V|}\sum_{t=1}^{|v|}\sum_{j=t,j\neq t}^{t+c} log[P(w_j|w_t)] \tag{10}$$

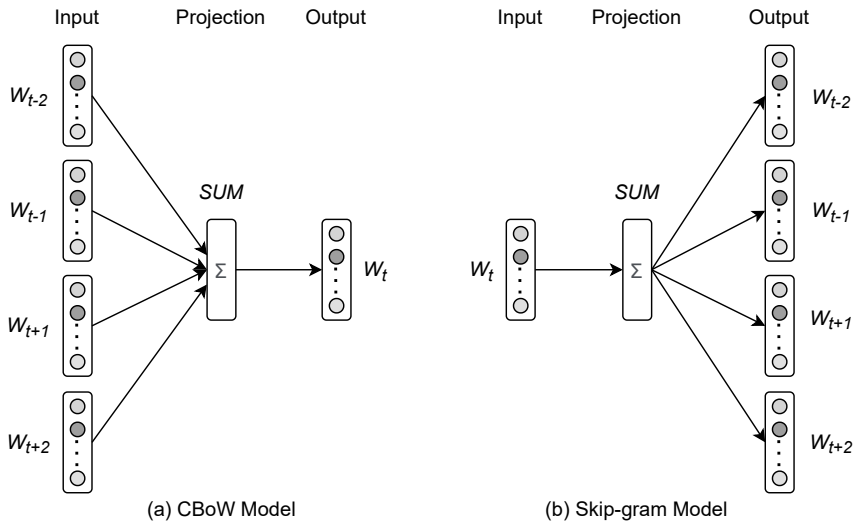

Figure 5: Model architecture (a) CBOW (b) Skip-gram.

As shown in Figure 5, CBoW deduces a target word when the context is given, whereas, when an input word is given, Skipgram infers the context words. Both learning algorithm architectures have three layers; input, projection and output layer, although their output formation process are divergent. During training process, CBoW works faster than skip-gram and reports slightly better for frequent words, while skip-gram represents infrequent words and phrases better even when there is less training data (Mikolov et al., 2013a). The feature vectors are attained from hidden layers against the input.

In this work, the Word2Vec model has been trained on a Shahmukhi corpus of 14.9 million words as described in Section 5.4. For training Word2Vec model, we used a python library named Genism[9]. The trained model had a vocabulary size of 73 thousands with minimum frequency of five. Each word had a dimension of 100. The window size was set to five, whereas default number of iterations were used. Shahmukhi being morphologically-rich language has many derived forms of a single root word which may produce OOV words. The limitation of

[9]https://pypi.org/project/gensim/

Word2Vec is their inability to represent polysemous words. An average vector was calculated for OOV words. Word2Vec model is unable to learn different contexts of a single word. Consequently, it places the final vector anywhere in the weighted middle of all words' connotations.

6.1.2. Contextualized Word Representations

In natural languages, words bear different meanings based on their contexts. Embeddings from Language Models (ELMo) embeddings (Peters et al., 2018) are one of the state of the art pre-trained transfer learning models which instigate contextualized embedding vectors. It extends the traditional word embedding models with bidirectional features using character convolutions. It performs proficiently for various NLP tasks as compared to other types of word embedding models such as Word2Vec (Mikolov et al., 2013b), fastText (Wang et al., 2018) and GloVe (Pennington et al., 2014).

From character sequences of words, ELMo learns morphology quite well and elucidate the word polysemy issues. Figure 6 shows Elmo architecture. ELMo embeddings architecture contains three neural network layers which are faster to train and adaptable to explicit tasks. ELMo model's first convolutional neural network layer operates on character level and works context independently. The convolutional layer is followed by two context dependent bidirectional layers where each bidirectional layer has two concatenated LSTMs. ELMo is able to knob OOV words (Ulčar & Robnik-Šikonja, 2019). The pre-trained ELMo weights are used to compute the feature vectors for all words in a sentence. These vectors are computed on the bases of their context words in a sentence. The contextualization is helpful to perform word sense disambiguation and further improves the NE labeling results. We have used 14.9 million Shahmukhi corpus to train ELMo embeddings. The trained model had a vocabulary size of 72 thousands with a dimension size of 128 for each word. The third layer of ELMo embeddings has been used in all of our experiments.

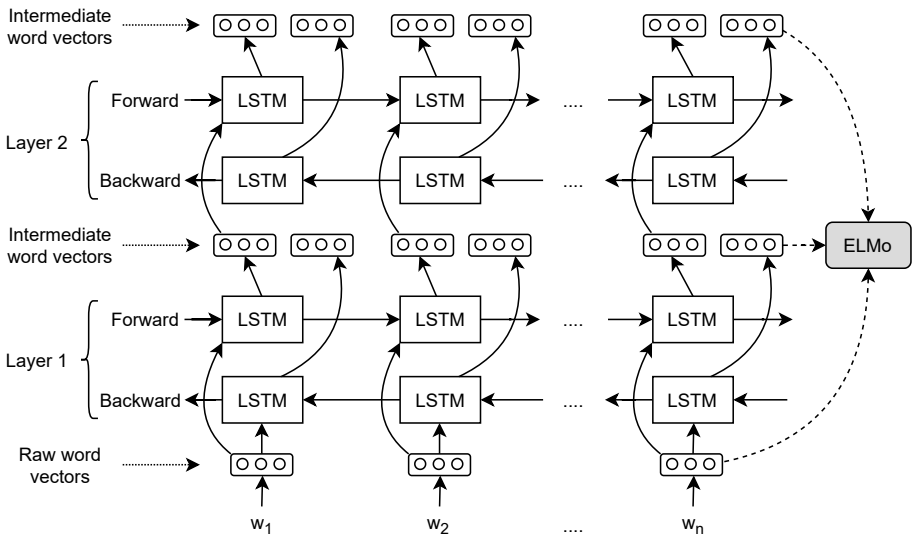

Figure 6: ELMo Architecture.

6.1.3. Analysis of Word Representations

To demonstrate the difference between context-free and contextualized word representations, we have performed an analysis of both types of embeddings on a tiny corpus containing two sentences as shown by (8) and (9).

(8) DrIvar=nE gaDDI xAlI saRak=tE buhat tEz
driver.M.Sg=Erg vehicle.F.Sg empty road.F.Sg=Loc very fast
calAI
drive.Perf.F.Sg
'The driver drove the car very fast on the empty road.'

(9) hazAr rOpE=dE nOTAN=dI gaDDI naIN labH
thousand rupee.M.Pl=Gen note.M.Pl=Gen bundle.F.Sg Neg find
rahI
live.Pres.F.Sg
'Cannot find the bundle of thousand rupee notes.'

To perform semantic analysis, we have selected a Shahmukhi word *gaDDI* which has meanings of 'vehicle' and 'bundle' in the language. We have analyzed

the sense representations with word enbeddings achieved from Word2Vec and ELMo. The first sentence (8) is about driving a vehicle on an empty road by a driver. However, the word *gaDDI* refers to several types of vehicles like car, bus, train etc. The second sentence (9) is about a lost bundle of money having currency notes of thousand rupees. The second sense of the word *gaDDI* gives meaning of bundle, pile etc. From the first sentence, we obtained the embedding vectors of *gaDDI*, *DrIvar* 'driver' and *saRak* 'road'. On the other hand, the words *gaDDI*, *nOTAN* 'notes' and *rOpE* 'rupees' have been selected from the second sentence. We further computed cosine similarities of related words with the pivotal word *gaDDI*. Table 6.1.3 shows results of cosine similarities for both types of word embeddings.

Table 3: Cosine similarities of word vectors obtained from Word2Vec and ELMo representations against the sample tiny corpus.

	Word2Vec		ELMo	
Word Vectors	*gaDDI* 'vehicle'	*gaDDI* 'bundle'	*gaDDI* 'vehicle'	*gaDDI* 'bundle'
DrIver 'driver'	0.5463204	0.5463204	**0.59956769**	0.39718021
saRak 'road'	0.6669706	0.6669706	**0.76936405**	0.50190787
nOTAN 'notes'	0.2107594	0.2107594	0.41038845	**0.64986019**
rOpE 'rupees'	0.2692004	0.2692004	0.28676091	**0.56692915**

Table 6.1.3 shows cosine similarities of *gaDDI* with four other words *DrIver* 'driver', *saRak* 'road', *nOTAN* 'notes' and *rOpE* 'rupees'. Results show that Word2Vec embeddings have a single sense for the word *gaDDI* which is 'vehicle' as it has higher similarities with *DrIver* and *saRak*. The sense of the word remains same even from a different context in sentence (9). Despite different meanings, Word2Vec representations are able to learn only one meaning of a word. On the other hand, ELMo embeddings are quite capable of learning contextualized word representations as shown in the Table 6.1.3. The word *gaDDI* has more similarity with words *DrIver* 'driver' and *saRak* 'road' from the first sentence (8). In the other context, the word *gaDDI* is more similar to

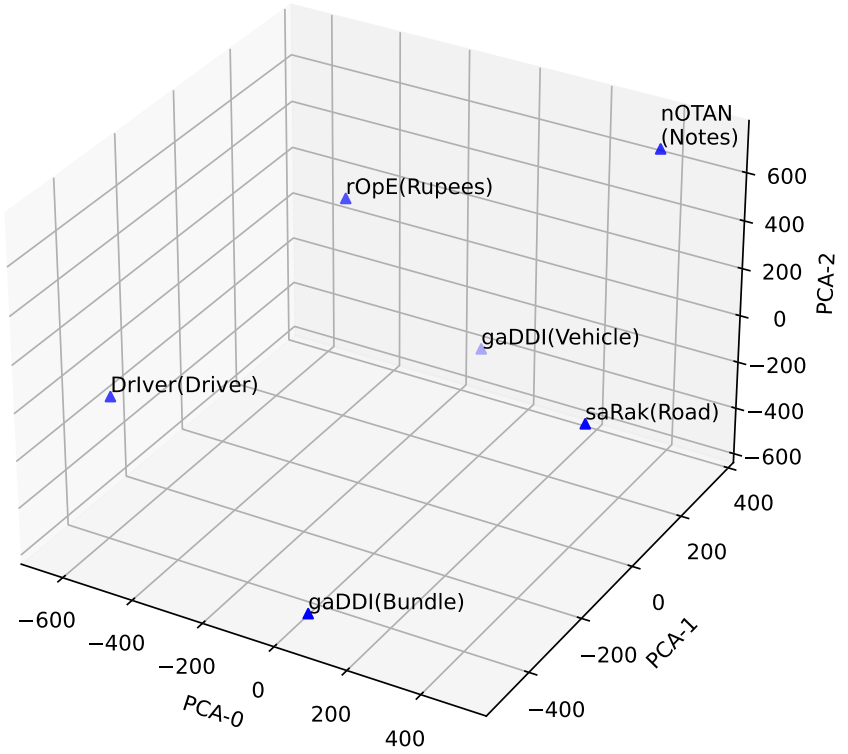

Figure 7: Word vector representations from ELMo representations based on their contexts.

the words *nOTAN* 'notes' and *rOpE* 'rupees'. From this example, it is quite clear that contextualized word embeddings ELMo are capable to learn multiple senses based on their contexts. Figure 6.1.3 further visualizes two different meanings of *gaDDI*.

Contextualized word vectors from ELMo embeddings have been plotted in a 3-dimensional plan in Figure 6.1.3. Principle Component Analysis (PCA) has been used to reduce dimensions from 128 to three dimensions. It is quite clear that the word *gaDDI* has two different meanings associated to it. The first meaning of *gaDDI* is 'vehicle' which also shows its similarity with related words *saRak* 'road' and *DrIver* 'driver'. Furthermore, the second meaning of *gaDDI* is 'bundle' which is being shown at a distance from the first vector of the word

and related words *nOTAN* 'notes' and *rOpE* 'rupees'. ELMo representations are capable to learn contextualized word embeddings and are helpful to improve performance of NLP tasks.

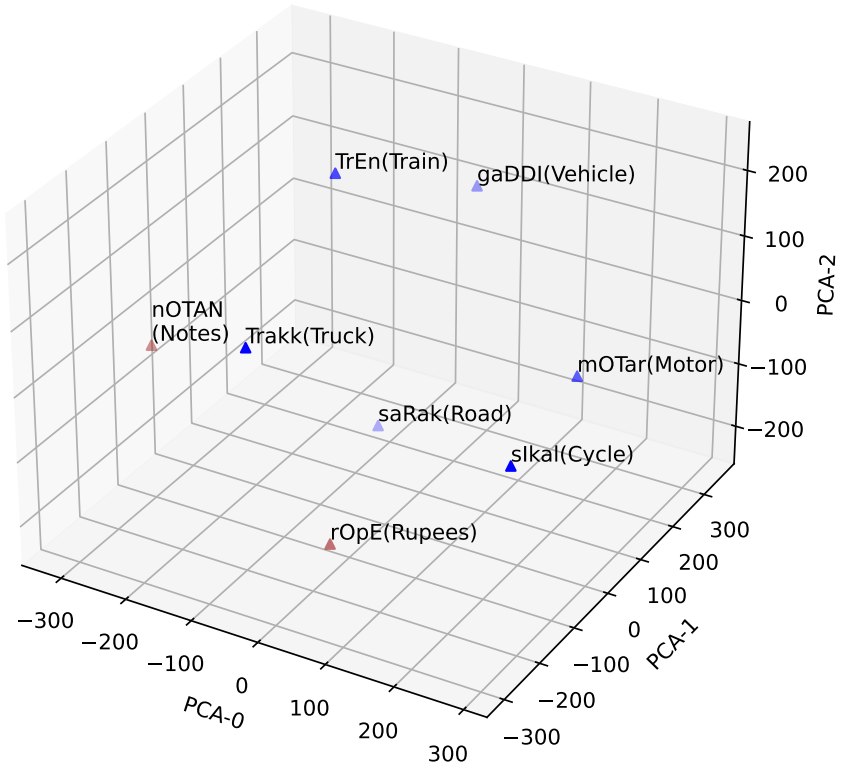

Figure 8: Word vector representations of similar words of *gaDDI* from Word2Vec embeddings.

On the contrary, Word2Vec produces context-free word representations by producing a single vector which in return associates a single meaning with each word. Figure 6.1.3 further shows top five similar words of *gaDDI* and two words from second sentence from the sample tiny corpus. The word *gaDDI* and its similar words are shown in blue color while other two in brown. The words with similar meanings are creating a cluster in the graph for words, *gaDDI* 'vehicle', *mOTar* 'motor', *Trakk* 'truck', *TrEn* 'train', *saRak* 'road', *sIkal* 'cycle'. The two words *nOTAN* 'notes' and *rOpE* 'rupees' have lower cosine similarity with

gaDDI and are plotted at a distance from the blue cluster. The figure is quite evident that Word2Vec embeddings produces context-free word embeddings for each word. However, it is capable to learn similar words and syntactic information of words in a corpus which is helpful to perform transfer learning. In our experiments, we have used both types of word representations to analyze their effect on named entity recognition for Shahmukhi.

7. Results and Discussion

After preprocessing, the number of tokens have been increased in the corpus because several punctuation and symbols have become separate tokens. However, the number of named entities remained same. Table 4 shows the details of the train and test sets. The dataset has been divided into two portions by using a 70:30 ratio. The train set contains 238,776 tokens and 11,562 named entities whereas the test set contains 100,778 tokens and 4,735 named entities.

Table 4: Train and test set statistics.

Category	Tokens	Named Entities	PER	ORG	LOC
Train set	238,776	11,562	7,832	1,406	2,324
Test set	100,778	4,735	3,314	606	815
Total	339,554	16,297	11,146	2,012	3,139

For NER experiments, we have used two types of labeling methods, IO and IOB. IO labeling annotates the tokens which are part of any NE by using I label and O label is used for out side of any NE. It reduces the number of labels but does not differentiate between two consecutive NEs. On the other hand, IOB labeling defines the beginning of an NE with label B and inside of an NE with label I. It identifies the NEs more accurately which may span over more than one tokens. However, by using the IO labeling, the evaluation is performed on the basis of single tokens.

The results of all experiments are presented in terms of the labeling accuracy, precision, recall and f-score in Table 5 and Table 7. Table 5 demonstrates results

for IO NER whereas Table 7 presents IOB NER results. All these results are reported on an unseen corpus because the word embeddings have been trained on a plain text corpus and the test set has been used for evaluation purposed only. Figure 7 further shows train and validation accuracy and loss for the best performing neural model on IO labels. The graphs in the figure have been plotted against 16 epochs but optimal number of epochs have been selected between six to ten epochs to avoid over-fitting.

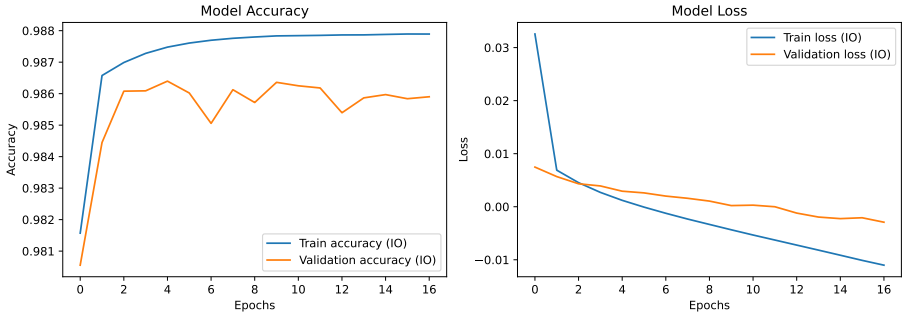

Figure 9: Train and validation accuracy and loss by using IO labels.

For IO based NER tagging, experiments were initiated with one BiLSTM layer. With merely a single BiLSTM layer, NER model performed with 97.89% accuracy and 75.52 f-score. Experiments were extended with POS tagging, character embeddings, pre-trained Word2Vec model and ELMo embeddings. The results with corresponding additional features are shown in Table 5. In the sixth experiment, we integrated one BiLSTM layer with character embeddings and POS feature. It performed with an accuracy of 98.18% and 78.51 f-score. After that, by adding Word2Vec vectors with these features, results were further improved. The POS tagging improved the results but using the word embedding that learns the syntactic and semantic information. IO NER model performed best with one hidden BiLSTM layer incorporating Word2Vec and ELMo embeddings together. It performed with 98.55% accuracy and 83.27 f-score.

The BiLSTM model along with *softmax* non-linearity and word representations performs quite well on IO labeling scheme. However, the experiments

Table 5: Results of IO NER models for all experiments conducted with different features. The proposed NER model outclassed with features that are bold.

Sr#	NER Model	Accuracy	Precision	Recall	F-score
1	1-BiLSTM	97.89	76.07	74.99	75.52
2	1-BiLSTM + Ch_emb	98.07	78.28	76.98	77.62
3	1-BiLSTM + Ch_emb + W2V	98.49	84.39	80.29	82.29
4	2-BiLSTM + Ch_emb + W2V	98.46	81.48	83.73	82.59
5	3-BiLSTM + Ch_emb + W2V	98.54	83.72	82.53	83.12
6	1-BiLSTM + Ch_emb + POS	98.18	80.51	76.62	78.51
7	1-BiLSTM + Ch_emb + W2V + POS	98.52	83.72	81.81	82.76
8	2-BiLSTM + Ch_emb + W2V + POS	98.46	80.48	85.07	82.71
9	1-BiLSTM + ELMo	98.42	81.24	82.23	81.73
10	1-BiLSTM + W2V + ELMo	98.55	83.7	82.85	83.27
11	2-BiLSTM + W2V + ELMo	98.48	83.14	81.37	82.25
12	1-BiLSTM + W2V + ELMo + Ch_emb	98.52	85.57	79.66	82.51
13	2-BiLSTM + W2V + ELMo + Ch_emb	98.54	85.3	80.16	82.65
14	1-BiLSTM-CRF + W2V + ELMo	98.56	83.62	83.27	83.45
15	2-BiLSTM-CRF + W2V + ELMo	98.46	80.37	85.11	82.67
16	1-BiLSTM-ATT-CRF + W2V + ELMo	98.44	87.28	75.54	80.99
17	2-BiLSTM-ATT-CRF + W2V + ELMo	98.50	85.33	79.15	82.13
18	CNN-BiLSTM-CRF	98.12	79.43	76.93	78.16
19	CNN-BiLSTM-CRF + W2V	98.43	83.64	79.91	81.73
20	CNN-BiLSTM-CRF + ELMo	98.44	87.57	75.03	80.82
21	**CNN-BiLSTM-CRF + W2V + ELMo**	**98.60**	**84.26**	**83.25**	**83.75**

have been further enhanced by adding more components in the model including CRF based output layer, self attention (ATT) and character Convolutional Neural Network (CNN) layers. By adding a CRF layer along with Word2Vec and ELMo embeddings, the model achieved an f-score of 83.45 showing an improvement of 0.18 score. Self attention layer was not much helpful to achieve improved

results. The experiments have been further enhanced by training CNN-LSTM-CRF model. The model has character convolution layer which helps to learn morphological information. The CNN-LSTM-CRF without additional features achieved an f-score of 78.16. However, pre-trained Word2Vec and ELMo representations improved the model's performance and achieved an f-score of 83.75 with an improvement of 0.3 points. The f-score for LOC, ORG and PER are 77.97, 69.17 and 87.81 respectively as shown in Table 7.

Table 6: Entity-wise results for IO labeling.

Sr#	Named Entity	Precision	Recall	F-score
1	LOC	76.94	79.02	77.97
2	ORG	71.48	67.00	69.17
3	PER	88.36	87.26	87.81

Table 7 shows entity-wise NER results which have been trained and achieved by using IO labeling scheme. The highest f-score has been achieved for the person entity which is 87.81. The location and organization entities have been recognized with f-scores of 77.97 and 69.17 respectively. For result analysis, Figure 7 shows confusion matrix for three named entities. The vertical axis shows actual named entities while the horizontal axis has predicted entities. IO labeling scheme considers single token entities. person entity has been recognized 2,891 times out of 2,931 entities. person has been recognized wrongly to location 22 times and to organization by 18 times. Similarly, the location entity has been predicted correctly by 644 times and 25 times as person and nine times as organization. The organization entity has been correctly recognized by 406 times and 17 and 29 times to person and location respectively. The Outside (O) label has been ignored in the confusion matrix.

Table 7 presents results for several NER neural models against IOB labels. Figure 7 shows accuracy and loss against train and validation sets for the best performing model. Graphs have been plotted for 16 epochs, however, optimal number of epochs (6-10) have been selected for experiments to ovoid over-fitting.

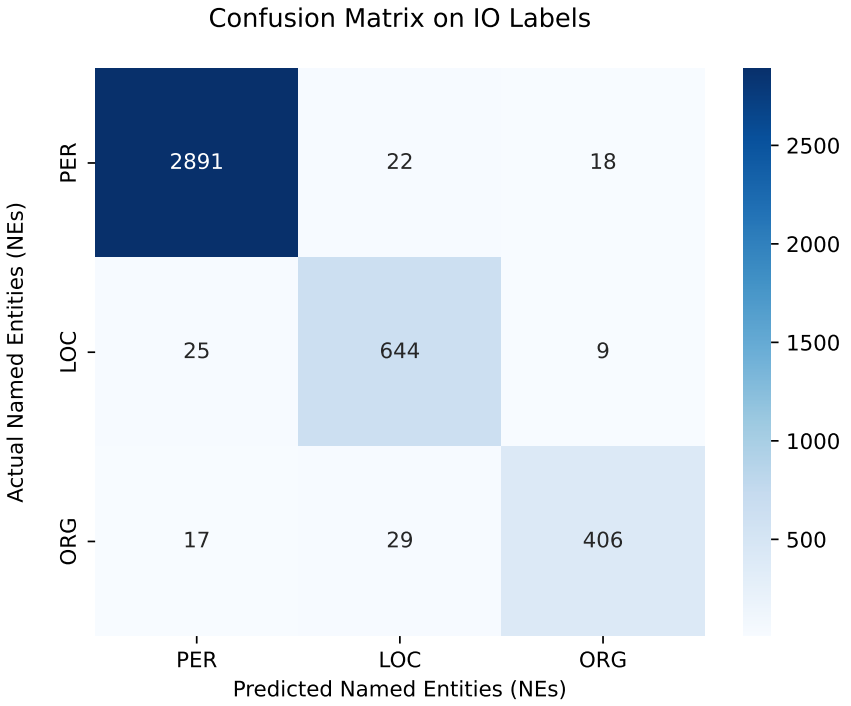

Figure 10: Confusion matrix for three named entities trained on IO labeling.

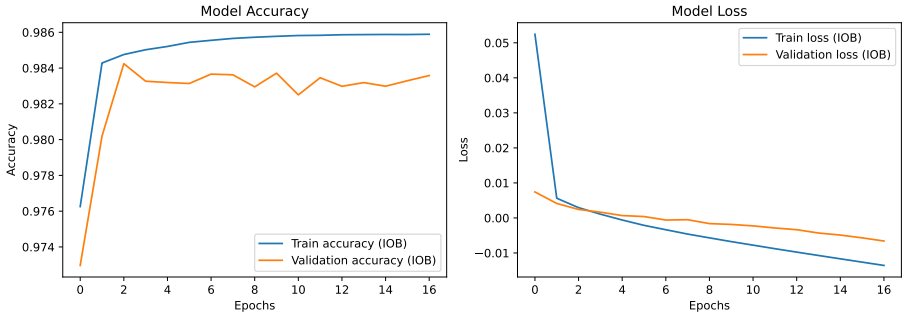

Figure 11: Train and validation accuracy and loss by using IOB labels.

IOB based NER models attained highest 75.55 f-score using two BiLSTM layers, Word2Vec and ELMo embeddings. With these features, an accuracy of 98.43% was achieved. Accuracy was highest 98.44% with one and two BiLSTM layers incorporating character, Word2Vec and ELMo embeddings. POS tagger model

Table 7: Results of IOB NER models for all experiments conducted with different features. The proposed NER model outclassed with features that are bold.

Sr#	NER Model	Accuracy	Precision	Recall	F-score
1	1-BiLSTM	97.74	62.91	67.56	65.15
2	1-BiLSTM + Ch_emb	97.81	62.73	70.51	66.39
3	1-BiLSTM + Ch_emb + W2V	98.35	71.63	73.85	72.72
4	2-BiLSTM + Ch_emb + W2V	98.37	72.76	73.6	73.18
5	3-BiLSTM + Ch_emb + W2V	98.32	74.11	70.02	72.0
6	1-BiLSTM + Ch_emb + POS	97.94	65.61	70.76	68.09
7	1-BiLSTM + Ch_emb + W2V + POS	98.33	72.42	71.25	71.83
8	2-BiLSTM + Ch_emb + W2V + POS	98.28	70.36	75.5	72.84
9	1-BiLSTM + ELMo	98.34	71.34	72.72	72.03
10	1-BiLSTM + W2V + ELMo	98.43	73.23	73.25	73.24
11	**2-BiLSTM + W2V + ELMo**	**98.43**	**74.01**	**77.15**	**75.55**
12	1-BiLSTM + W2V + ELMo + Ch_emb	98.44	72.86	74.06	73.45
13	2-BiLSTM + W2V + ELMo + Ch_emb	98.44	74.41	75.01	74.71
14	1-BiLSTM-CRF + W2V + ELMo	98.41	74.18	75.92	75.04
15	2-BiLSTM-CRF + W2V + ELMo	98.37	72.16	77.54	74.75
16	1-BiLSTM-ATT-CRF + W2V + ELMo	98.38	76.86	72.72	74.73
17	2-BiLSTM-ATT-CRF + W2V + ELMo	98.24	69.74	77.68	73.50
18	CNN-BiLSTM-CRF	97.89	68.07	66.47	67.26
19	CNN-BiLSTM-CRF + W2V	98.28	74.36	70.93	72.60
20	CNN-BiLSTM-CRF + ELMo	98.43	78.99	71.49	75.06
21	CNN-BiLSTM-CRF + W2V + ELMo	98.46	76.61	73.11	74.82

was assimilated for IOB models as well but it was observed that ELMo embeddings significantly improved the results as compared to POS feature. POS tags are helpful to improve NER scores when used without word embeddings. Transfer learning based NER models perform well as compared to standalone models.

Table 8: Entity-wise results for IOB labeling.

Sr#	Named Entity	Precision	Recall	F-score
1	LOC	74.85	74.85	74.85
2	ORG	59.48	57.74	58.6
3	PER	75.36	80.29	77.74

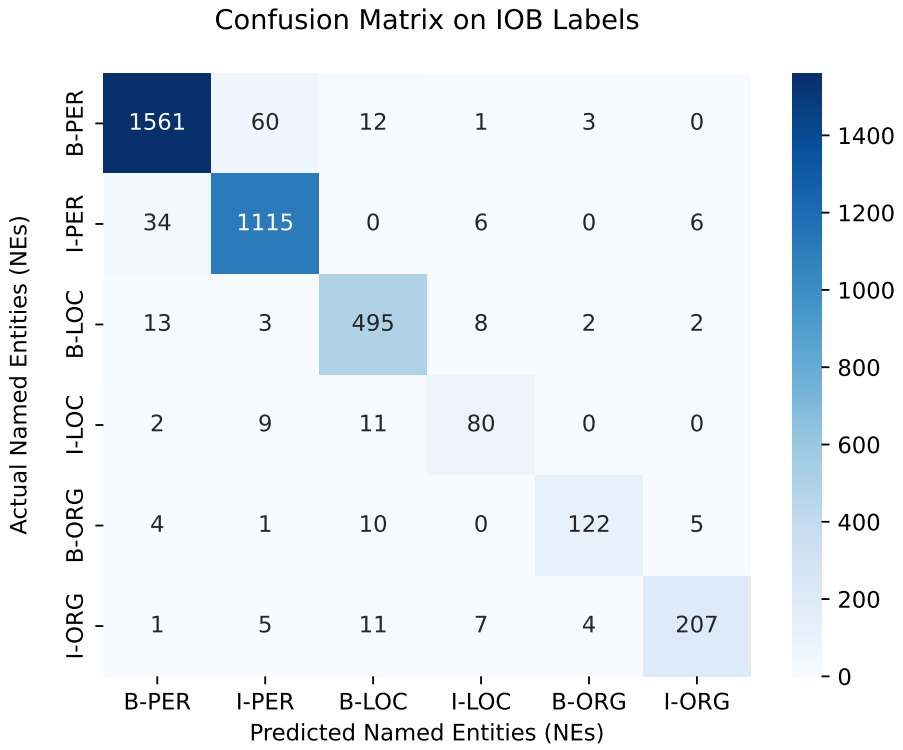

Figure 12: Confusion matrix for named entities trained on IOB labeling.

Beside bidirectional LSTM networks and embedding vectors, we further developed and trained different architectures for IOB labeling scheme by adding CRF output layer, self attention and character convolution layer. However, pre-trained word embeddings also have been used in additional architectures. The BiLSTM with CRF output layer achieved an f-score of 75.04. The self attention

layer was not helpful to improve the results. The CNN-BiLSTM-CRF model has been trained by using Word2Vec and ELMo representations and achieved best f-score of 75.06. However, the BiLSTM model with *softmax* non-linearity, Word2Vec and ELMo embedding still performed better with an f-score of 75.55. For each named entity, LOC, ORG and PER, it performed with 74.85, 58.6 and 77.74 f-scores respectively as shown in Table 8.

Furthermore, Figure 7 presents the result analysis by showing a confusion matrix achieved by using IOB labeling scheme. The matrix shows six labels which have prefix B and I. The B-PER label shows higher confusions with I-PER as they both are used to annotate persons. In the same way, I-PER has more confusions with B-PER label. B-LOC label has confusions with I-LOC and B-PER. I-LOC label shows confusions with B-LOC due to similar entities. The B-ORG and I-ORG have been wrongly recognized more to B-LOC as compared to other labels. The IOB labels achieve lower f-score because the NEs span multiple words. However, this labeling scheme accurately identifies distinct as well as consecutive entities of same or different types. The label O, which has been used to annotate all outside words, has not been included in the confusion matrix.

Ahmad et al. (2020) used five supervised learning techniques to illustrate usability of their developed Shahmukhi NER corpus. RNN based NER model performed with an f-score of 85.2. Though, the reported results were quite significant but their work has limitations which are addressed in our work. They trained word embeddings on the whole dataset then splitted it into train and test sets which makes their results invalid and biased. On the other hand, they have just experimented with IO annotation scheme which produced higher scores but it can not differentiate between consecutive named entities of same types.

The techniques in our research offer several benefits as compared to the existing work. We developed the neural models by using bidirectional long-short term memory networks which are considered better performing models for sequence labeling problems like named entity recognition. We further integrated different features like character embeddings, POS tagging and transfer learning.

Transfer learning improved the NER results significantly. We also produced a cleaned dataset and valid results by performing evaluations on an unseen test corpus. We carried the experiments on both annotation schemes, IO and IOB, and reported their effect on the results. Conclusively, this is first research to predict named entities for Shahmukhi language by using POS tagging and transfer learning.

Shahmukhi is a an under-resourced language which lacks essential computational resources. The main limitation of this work is the size of the dataset which is annotated with three named entities. The transfer learning has been performed by training word embeddings on a corpus of 14.9 million words which is quite less as compared to resource-rich languages. In future work, the dataset can be further annotated to have additional named entities like date, time, number, designation etc. The results can further be enhanced by developing a larger corpus to perform transfer learning.

In this section, NER results have been presented against IO and IOB labels. The IO labels annotate single word named entities and hence produce higher results. On the other hand, the IOB scheme annotates NEs spanning multiple words. There are a few limitations and challenges of this research. First of all, the dataset has been annotated with three named entities which are person, location and organization. Therefore, most of the tokens have been annotated with outside O label. All these NEs are nouns and hence tagged with two POS tags, NN for common nouns and NNP for proper nouns. There are no additional tags to annotate number and gender of nouns. This is the main reason that the results were not enhanced much by using POS tags as feature in the neural models. In literature, there are also some other named entities like designations, numbers, date and time. All these NEs are represented with different POS tags in the tag set. In future, the dataset set can be enhanced to have additional NEs.

Furthermore, some linguistic properties also make it challenging to learn NEs, which are, word segmentation and lack of capitalization. The word segmentation problem in Shahmukhi script may produce different tokenization hence

different annotation. For example, the person name *muhammad sAem* can also be written without space between two tokens. Zero-Width-Non-Joiner (ZWNJ) is also used in the digital Shahmukhi text to ovoid character joining. This property of the language produces different tokenization for same NEs and hence may reduce the NER model performance. There are no capitalization in Shahmukhi as compared to English and some other Western languages. Therefore, proper nouns are treated just like common nouns in the corpus.

Finally, to achieve higher NER results, larger datasets are required. Shahmukhi is a low-resourced South-Asian language which lacks essential data resources. Beside the annotated datasets, transfer learning is a suitable option for these type of languages. To enhance the performance of our models, we trained word representations on a corpus of 14.9 words. The corpus has been downloaded from all valid sources available online as per best of our knowledge. The models' performance can be enhanced by developing larger datasets for Shahmukhi NER and transfer learning in future.

8. Conclusion

This article presents four key contributions for the advancement of the Shahmukhi NER. First, we prepared the Shahmukhi NER corpus by executing Unicode normalization and cleaning. Second, experiments have been performed by incorporating various features with BiLSTM NER taggers such as character embeddings, POS tagging and transfer learning. Word2Vec and ELMo embeddings have been trained on a plain Shahmukhi corpus of 14.9 million words. POS tags improved the NER scores, but word embedding models outperformed it. ELMo embeddings significantly improved the results by prompting contextualized embedding vectors. Third, we studied the impact of IO and IOB annotation schemes on Shahmukhi. Fourth, unseen corpus has been used for models evaluation. IO based NER model performed with an accuracy of 98.60% and 83.75 f-score. IOB based NER model performed with an accuracy of 98.43% and 75.55 f-score. IO based NER models outperformed IOB models, but IO annotation

scheme does not correctly encode consecutive named entities of the same type, whereas IOB scheme can identify them accurately. In future, Shahmukhi NER system accuracy could be improved by using large Shahmukhi NER corpora with more types of named entities and accurate tokenization. Experiments can be extended to study the impact of other annotations schemes. This work will assist as a milestone for various Shahmukhi NLP tasks such as IR, machine translation and text summarization.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China under the Grant No. 62072409.

References

Ahmad, M. T., Malik, M. K., Shahzad, K., Aslam, F., Iqbal, A., Nawaz, Z., & Bukhari, F. (2020). Named Entity Recognition and Classification for Punjabi Shahmukhi. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, *19*, 1–13.

Ahmed, T., Ehsan, T., Ashraf, A., u Rahman, M., Hussain, S., & Butt, M. (2020). A Multilayered Urdu Treebank. *LANGUAGE & TECHNOLOGY*, .

Ahmed, T., Urooj, S., Hussain, S., Mustafa, A., Parveen, R., Adeeba, F., Hautli, A., & Butt, M. (2015). The CLE Urdu POS tagset. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation* (pp. 2920–2925).

Alshammari, N., & Alanazi, S. (2020). An Arabic Dataset for Disease Named Entity Recognition with Multi-Annotation Schemes. *Data*, *5*, 60.

Baig, A., Rahman, M. U., Kazi, H., & Baloch, A. (2020). Developing a POS Tagged Corpus of Urdu Tweets. *Computers*, *9*, 90.

Bhurgari, A. (2007). Enabling Pakistani Languages through Unicode, published at http://download. microsoft. com/download/1/4/2/142aef9f-1a74-4a24-b1f4-782d48d41a6d. *PakLang. pdf*, .

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Chopra, D., & Morwal, S. (2012). Named Entity Recognition in Punjabi using Hidden Markov Model. *International Journal of Computer Science & Engineering Technology*, *3*, 616–620.

Cowan, B., Zethelius, S., Luk, B., Baras, T., Ukarde, P., & Zhang, D. (2015). Named Entity Recognition in Travel-Related Search Queries. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (pp. 3935–3941).

Deng, L. (2014). A Tutorial Survey of Architectures, Algorithms, and Applications for Deep Learning. *APSIPA Transactions on Signal and Information Processing*, *3*.

Du, J., Vong, C.-M., & Chen, C. P. (2020). Novel Efficient RNN and LSTM-like Architectures: Recurrent and Gated Broad Learning Systems and their Applications for Text Classification. *IEEE transactions on cybernetics*, *51*, 1586–1597.

Ehsan, T., & Asif, H. S. (2018). Finding Topics in Urdu: A Study of Applicability of Document Clustering in Urdu Language. *Pakistan Journal of Engineering and Applied Sciences*, .

Ehsan, T., & Butt, M. (2020). Dependency Parsing for Urdu: Resources, Conversions and Learning. In *Proceedings of the 12th Language Resources and Evaluation Conference* (pp. 5202–5207).

Ehsan, T., & Hussain, S. (2019). Analysis of Experiments on Statistical and Neural Parsing for a Morphologically Rich and Free Word Order Language Urdu. *IEEE Access*, *7*, 161776–161793.

Ehsan, T., & Hussain, S. (2020). Development and Evaluation of an Urdu Treebank (CLE-UTB) and a Statistical Parser. *Language Resources and Evaluation*, (pp. 1–40).

Ehsan, T., & Hussain, S. (2022). *Statistical Parser for Urdu*. Ph.D. dissertation University of Engineering and Technology, Lahore, Pakistan.

Ehsan, T., Khalid, J., Ambreen, S., Mustafa, A., & Hussain, S. (2022). Improving Phrase Chunking by using Contextualized Word Embeddings for a Morphologically Rich Language. *Arabian Journal for Science and Engineering*, (pp. 1–19).

Goldberg, Y., & Levy, O. (2014). Word2Vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. *arXiv preprint arXiv:1402.3722*, .

Goyal, A. (2008). Named Entity Eecognition for South Asian Languages. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*.

Hasan, E., Iqbal, M. M., Azeemi, Q. R., & Javeed, A. (2015). An online Punjabi Shahmukhi Lexical Resource. *Sci. Int (Lahore)*, *27*, 2529–2535.

Humayoun, M., & Ranta, A. (2010). Developing Punjabi Morphology, Corpus and Lexicon. In *Proceedings of the 24th Pacific Asia conference on language, information and computation* (pp. 163–172).

Jahangir, F., Anwar, W., Bajwa, U. I., & Wang, X. (2012). N-gram and Gazetteer List based Named Entity Recognition for Urdu: A Scarce Resourced Language. In *Proceedings of the 10th Workshop on Asian Language Resources* (pp. 95–104).

Kanwal, S., Malik, K., Shahzad, K., Aslam, F., & Nawaz, Z. (2019). Urdu Named Entity Recognition: Corpus Generation and Deep Learning Applications. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, *19*, 1–13.

Kaur, A., & Josan, G. S. (2014). Evaluation of Punjabi Named Entity Recognition using Context Word Feature. *International Journal of Computer Applications*, *96*.

Kaur, A., & Josan, G. S. (2015). Evaluation of Named Entity Features for Punjabi Language. *Procedia Computer Science*, *46*, 159–166.

Khademi, M. E., & Fakhredanesh, M. (2020). Persian Automatic Text Summarization based on Named Entity Recognition. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, (pp. 1–12).

Khan, W., Daud, A., Alotaibi, F., Aljohani, N., & Arafat, S. (2020). Deep Recurrent Neural Networks with Word Embeddings for Urdu Named Entity Recognition. *ETRI Journal*, *42*, 90–100.

Khan, W., Daud, A., Khan, K., Nasir, J. A., Basheri, M., Aljohani, N., & Alotaibi, F. S. (2019). Part of Speech Tagging in Urdu: Comparison of Machine and Deep Learning Approaches. *IEEE Access*, *7*, 38918–38936.

Liu, Y., Wang, J., Li, J., Niu, S., Wu, L., & Song, H. (2021). Zero-Bias Beep-Learning-Enabled Quickest Abnormal Event Detection in IoT. *IEEE Internet of Things Journal*, *9*, 11385–11395.

Liu, Y., Wang, J., Li, J., Song, H., Yang, T., Niu, S., & Ming, Z. (2020). Zero-Bias Deep Learning for Accurate Identification of Internet-of-Things (IoT) Devices. *IEEE Internet of Things Journal*, *8*, 2627–2634.

Mahdaouy, A. E., Gaussier, E., & Alaoui, S. O. E. (2016). Arabic text classification based on word and document embeddings. In *International conference on advanced intelligent systems and informatics* (pp. 32–41). Springer.

Malik, M. K. (2017). Urdu Named Entity recognition and Classification System using Artificial Neural Network. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, *17*, 1–13.

Malik, M. K., Ahmed, T., Sulger, S., Bögel, T., Gulzar, A., Raza, G., Hussain, S., & Butt, M. (2010). Transliterating Urdu for a Broad-Coverage Urdu/Hindi LFG grammar. In *LREC 2010, Seventh International Conference on Language Resources and Evaluation* (pp. 2921–2927).

Malte, A., & Ratadiya, P. (2019). Evolution of Transfer Learning in Natural Language Processing. *arXiv preprint arXiv:1910.07370*, .

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Mollá, D., Van Zaanen, M., & Smith, D. (2006). Named Entity Recognition for Question Answering. In *Proceedings of the Australasian language technology workshop 2006* (pp. 51–58).

Mukund, S., Srihari, R., & Peterson, E. (2010). An Information-Extraction System for Urdu—A Resource-Poor Language. *ACM Transactions on Asian Language Information Processing (TALIP)*, *9*, 1–43.

Mukund, S., & Srihari, R. K. (2009). NE Tagging for Urdu based on Bootstrap POS Learning. In *Proceedings of the Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies (CLIAWS3)* (pp. 61–69).

Mumtaz, B., Urooj, S., Hussain, S., & Haq, E. U. (2016). Break Index (BI) Annotated Speech Corpus for Urdu TTS. In *2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of*

Speech Databases and Assessment Techniques (O-COCOSDA) (pp. 22–27). IEEE.

Niu, S., Liu, M., Liu, Y., Wang, J., & Song, H. (2021). Distant Domain Transfer Learning for Medical Imaging. *IEEE Journal of Biomedical and Health Informatics*, *25*, 3784–3793.

Niu, S., Liu, Y., Wang, J., & Song, H. (2020). A Decade Survey of Transfer Learning (2010–2020). *IEEE Transactions on Artificial Intelligence*, *1*, 151–166.

PBS (2017). Population by Mother Tongue Pakistan Bureau of Statistics. URL: `http://www.pbs.gov.pk/content/population-mother-tongue` (Accessed on 09/01/2020).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextualized Word Representations. *arXiv preprint arXiv:1802.05365*, .

Popovski, G., Seljak, B. K., & Eftimov, T. (2020). A Survey of Named-Entity Recognition Methods for Food Information Extraction. *IEEE Access*, *8*, 31586–31594.

Riaz, K. (2010). Rule-based Named Entity Recognition in Urdu. In *Proceedings of the 2010 named entities workshop* (pp. 126–135).

Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. *arXiv preprint cs/0306050*, .

Sekhar, S. M., Kashyap, G., Bhansali, A., Singh, K. et al. (2022). Dysarthric-Speech Detection using Transfer Learning with Convolutional Neural Networks. *ICT Express*, *8*, 61–64.

Sharma, S. K. (2016). Assigning the Correct Word Class to Punjabi Unknown Words using CRF. *International Journal of Computer Applications*, *975*, 8887.

Simons, G. F., & Fennig, C. D. (2017). Ethnologue: Languages of the World (20th edn.). Dallas, TX: SIL international.

Singh, A. K. (2008). Named Entity Eecognition for South and South East Asian Languages: Taking Stock. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*.

Singh, K. (2013). Name Entity Recognition on Punjabi Language. *International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)*, *3*, 95–102.

Singh, U., Goyal, V., & Lehal, G. S. (2012). Named Entity Recognition System for Urdu. In *Proceedings of COLING 2012* (pp. 2507–2518).

Tehseen, A., Ehsan, T., Liaqat, H. B., Ali, A., & Al-Fuqaha, A. (2022). Neural POS Tagging of Shahmukhi by Using Contextualized Word Representations. *Journal of King Saud University-Computer and Information Sciences*, . doi:https://doi.org/10.1016/j.jksuci.2022.12.004.

Ulčar, M., & Robnik-Šikonja, M. (2019). High Quality ELMo Embeddings for Seven Less-Resourced Languages. *arXiv preprint arXiv:1911.10049*, .

Urooj, S., Shams, S., Hussain, S., & Adeeba, F. (2014). Sense Tagged CLE Urdu Digest Corpus. *Centre for Language Engineering, Al-Khawarizmi Institute of Compute Science, University of Engineering and Technology, Lahore*, .

Vu, V.-H., Nguyen, Q.-P., Nguyen, K.-H., Shin, J.-C., & Ock, C.-Y. (2020). Korean-Vietnamese Neural Machine Translation with Named Entity Recognition and Part-of-Speech Tags. *IEICE Transactions on Information and Systems*, *103*, 866–873.

Wang, C., Cho, K., & Kiela, D. (2018). Code-Switched Named Entity Recognition with Embedding Attention. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (pp. 154–158).

Wang, J., Liu, Y., Niu, S., Jing, W., & Song, H. (2021). Throughput Optimization in Heterogeneous Swarms of Unmanned Aircraft Systems for Advanced Aerial Mobility. *IEEE Transactions on Intelligent Transportation Systems*, *23*, 2752–2761.