ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 12 N. 1 (2023), e31084
eISSN: 2255-2863
DOI: https://doi.org/10.14201/adcaij.31084

Comparison of Pre-trained vs Custom- trained Word Embedding Models for Word Sense Disambiguation

Muhammad Farhat Ullah[a], Ali Saeed[b], and Naveed Hussain[b]

[a] Department of Software Engineering, University of Lahore, Lahore, Pakistan
[b] Faculty of Information Technology, Department of Software Engineering, University of Central Punjab, Lahore, Pakistan
farhat.ullah@se.uol.edu.pk, ali.saeed@ucp.edu.pk
Corresponding Author: dr.naveedhussain@ucp.edu.pk

KEYWORDS

word sense disambiguation; deep learning approaches; word embedding approaches; lexical and all words sample

ABSTRACT

The prime objective of word sense disambiguation (WSD) is to develop such machines that can automatically recognize the actual meaning (sense) of ambiguous words in a sentence. WSD can improve various NLP and HCI challenges. Researchers explored a wide variety of methods to resolve this issue of sense ambiguity. However, majorly, their focus was on English and some other well-reputed languages. Urdu with more than 300 million users and a large amount of electronic text available on the web is still unexplored. In recent years, for a variety of Natural Language Processing tasks, word embedding methods have proven extremely successful. This study evaluates, compares, and applies a variety of word embedding approaches to Urdu Word embedding (both Lexical Sample and All-Words), including pre-trained (Word2Vec, Glove, and FastText) as well as custom-trained (Word2Vec, Glove, and FastText trained on the Ur-Mono corpus). Two benchmark corpora are used for the evaluation in this study: (1) the UAW-WSD-18 corpus and (2) the ULS-WSD-18 corpus. For Urdu All-Words WSD tasks, top results have been achieved (Accuracy=60.07 and F_1=0.45) using pre-trained FastText. For the Lexical Sample, WSD has been achieved (Accuracy=70.93 and F_1=0.60) using custom-trained GloVe word embedding method.

1. Introduction

Word sense disambiguation is an open challenge of computational linguistics. The focus of WSD is to design and develop such intelligent systems that can automatically understand the sense of ambiguous words. According to Navigli, the research on WSD started in 1940, however, the research community is still working on this topic (Dongsuk *et al.*, 2018; Le *et al.,* 2018; Wang *et al.,* 2020). From the year 2000 onward, most researchers attempted to solve WSD using machine-learning approaches with hand-crafted features (Broda *et al.,* 2013; lgen, Adali and Tantu, 2012). In general, from 2007 until now, deep neural networks with automatic features of word embedding (WE) have been predominant (Le *et al.,* 2018; Ali *et al.,* 2019; Wu *et al.,* 2015). However, the majority of work on WSD is for English and other well-reputed languages, and there have been few attempts for Urdu. In recent years, some researchers have worked on WSD, including (Mir *et al.,* 2023; Zhang, *et al.* 2023; Ramya and Karthik, 2023).

Earlier WSD research has highlighted two important roles: (1) Lexical Sample (or Targeted Words) WSD and (2) All-Words WSD (Cotton, Edmonds, and Scott 2001; Mihalcea, Chklovski, and Kilgarriff 2004). The purpose of a lexical sample or targeted words is to clarify the meaning of a group of words from a given text. The goal of All-Words WSD is to clarify the meaning of every ambiguous word that appears in a given text. Training a classifier for each target word is often the method employed for lexical sample tasks. This method is effective for developing WSD systems with high accuracy, however, it can only be used for the words that are being targeted and calls for annotated training data. However, the task is more difficult, as it is unable to gather training data for a very large lexical sample. However, solutions to the All-Words challenge are typically thought to be more beneficial for downstream applications. The Urdu language's All-Words are the focus of this research.

Urdu is one of the most widely used languages worldwide. An estimated 300 million people speak Urdu, which is spoken in 20 different countries (Sarmad, 2008; Khan *et al.*, 2016; Kashif, 2010). The primary cause of this enormous number is the widespread distribution of South Asians (Rahman 2004). In Pakistan, Bangladesh, India, and Jammu and Kashmir, Urdu is widely spoken. Its vocabulary and grammar are heavily influenced by Persian, Arabic, Turkish, and other South Asian languages (Rahman, 2004), and its verbs and nouns can have more than 40 different forms, making it quite challenging to understand (Hussain, Naseer and Sarmad, 2009). There are no corpora for the Urdu language, despite the fact that they are required to create, assess, and compare various WSD systems. In this work, the Urdu All-Words WSD task is performed using a benchmark corpus.

For South Asian languages, notably Urdu, where curated corpus resources are sorely inadequate, the issue of WSD has not been properly investigated (Rahman, 2004). The focus of this study is to identify the most successful word embedding model for both types of WSD. This study compares the outcomes of five pre-trained WE models i.e. Word2Vec (trained by Samar (Haider, 2018), Dr.Khurram (Kanwal *et al.,* 2019)) and FastText (trained by Facebook (Bojanowski, 2017), Dr.Khurram (Kanwal *et al.,* 2019)). And three custom WE models i.e. Word2Vec (Mikolov, 2013), GloVe (Pennington *et al.,* 2014), and FastText (Bojanowski, 2017) trained on Ur-Mono (Jawaid *et al.,* 2014). The performance of all WE models has been computed with five deep learning methods i.e., simple RNN, LSTM, GRU, bidirectional LSTM, and bidirectional GRU.

The remainder of this article is structured as follows: A survey of the current contributions made by various languages to the WSD task is presented in Section 2. Section 3 explains the detail of the word embedding layer and deep learning model description on the Urdu WSD (both Lexical Sample and All-Words) corpus. Section 4 explains the experimental setup. Section 5 shows the results and their analysis. Finally, Section 6 concludes the work presented in this article and outlines future research.

2. Related Work

Previously, several researchers proposed their solutions for WSD tasks in various languages. In 2015, a group of researchers used a semi-supervised approach to address the problem of WSD in the English language (Taghipour and Tou, 2015). The authors used WE which is likely to provide useful linguistic information. Adding continuous-space word representations can give useful information to the classifier, and the classifier can learn stronger discriminatory parameters based on that information. The authors used a form of word embeddings obtained from feed-forward neural networks. They have also proposed a new approach for applying discriminatory information to these embeddings once provided to a supervised WSD system. The system was evaluated on All-Words tasks, lexical sample tasks, domain specific dataset, and accuracy was improved consistently, and they achieved 68.2% accuracy.

In 2017, word embedding was used for the creation of a historical dictionary, this work examines the idea of using word embedding to address the issue of the word sense disambiguation problem. In this work, the authors proposed the method of measuring the semantic relationship between the meaning of the use of unclear words and the description of their senses. This approach is carried out by training the word vectors from the corpus using the Skip Gram model (Mikolov, 2013), and by describing the meaning of the word to be disambiguated and all the senses as vectors in multidimensional space. Cosine similarity is used to compare the similarity of the context vector with the target word sense vectors, disambiguated sense allocated the highest achieved similarity sense. Experiments showed an accuracy of 78% on 10 ambiguous words in Arabic (Rim *et al.,* 2017).

Another study on Hindi word sense disambiguation using word embedding by (Archana and Lobiyal, 2020) was carried out in 2020. The authors used Indowordnet for word senses. In this proposed system, they used already available semantic relations to map their target word, the closest proximity sensor was used to classify the sense of an ambiguous word. The authors first used Mikolov's word-2vec models to train the word vectors from the corpus. After that, the target words were shown in vector space with all their possible senses. At the end, with help of the cosine similarity of context vectors and sensing vectors, a score was measured. In the case of WSD highest, the similarity highest is the sense of the ambiguous word. 52% accuracy has been achieved on the test dataset during the evolution of the proposed model.

Moreover, another approach was created by (Uslu *et al.,* 2018), called fastSense. This model takes text as input and generates senses according to the input text. Its single hidden layer is an embedding layer that converts word indexes from the input layer to word vectors. This model was evaluated in the framework of Senseval and SemEval, and experiments were based on German Wikipedia. This model achieved a 0.81 F_1-score on Wikipedia-based data. According to the author of fastSense it works with huge datasets and surpasses state-of-the-art tools.

Although not enough work has been done for Urdu word sense disambiguation using word embedding, some researchers have attempted Urdu word sense disambiguation (Hussain, Naseer and Sarmad, 2009). The authors used Bayesian classification for lexical ambiguity of word sense, it is also called word sense disambiguation for Urdu words. In this article, the authors used four words in Urdu, three of them were verbs and one was a noun.

Moreover, another approach to addressing the problem of Urdu WSD was given by (Abid *et al.,* 2018), where the authors used already available linguistic knowledge such as parts of speech information. In this paper, the authors used several machine learning algorithms including Bayes, decision tree, and support vector machine. However, Bayes outperformed the other by achieving a 0.71 F_1-score.

In 2016, a group of researchers addressed the problem of Urdu WSD. The Urdu WSD task (Arif *et al.,* 2016) contributed with a freely available corpus. It consisted of 50 words, 30 were nouns, 11 were adjectives and 9 of them were verbs and it included 7,185 sentences. WSD approaches have been applied to the corpus for the suitability and evaluation of the corpus. The bag of words approach outperformed the other WSD experiments.

Above all, some researchers worked to address the problem of WSD on different corpora. In 2019, (Saeed *et al.,* 2019) developed a corpus consisting of 252 instances with 856 ambiguous words. By following the research community, the authors also used ML techniques to solve the problem of feature extraction. They used word n-gram and character n-gram. The authors applied similarity techniques on these features and realized that word 4 gram outperformed all feature extraction techniques and got an accuracy of 57.71%.

Table 1. Comprehensive overview of related work

Sr. No.	Reference and Year	Technique	Language	Dataset	Evaluation Measure and obtained Score
1.	(Taghipour and Tou, 2015)	WE with supervised WSD System	English	Lexical Sample	Accuracy =68.2%
2.	(Rim, *et al.* 2017)	WE skip-gram, Cosine similarity	Arabic	10 ambiguous words	Accuracy =78%
3.	(Archana and Lobiyal, 2020)	WE word2vec, Cosine similarity	Hindi	Indowordnet	Accuracy =52%
4.	(Uslu *et al.,* 2018)	fastSense	German	German Wikipedia data	F1-score = 0.81
5.	(Abid *et al.,* 2018)	Bayes	Urdu	Urdu WSD	F1-score= 0.71
6.	(Saeed *et al.,* 2019)	Word 4 gram, similarity	Urdu	All Words WSD	Accuracy = 57.71%
7.	(Das Dawn *et al.,* 2023)	CIM Random Forest	Bengali	100 Bengali polysemous words	Accuracy = 80%

Table 1 provides a comprehensive overview of previous work done by different researchers. Almost all researchers address the problem of WSD by handcrafted or ML techniques. More specifically, for UAW-WSD-18 and ULS-WSD-18, no one applied deep learning on top of state-of-the-art word embedding feature extraction techniques. This study focuses on addressing the Urdu WSD by applying deep learning on top of word embedding features.

3. Word Embedding Methods used for Urdu Word Sense Disambiguation

This section discusses the main contribution of the embedding layer in the deep neural network model. This study used four publicly available pre-trained embedding layers (Khurram Urdu word embeddings (Word2Vec, FastText), Samar Urdu word embedding (Word2Vec), FastText trained by

Facebook and additionally three custom trained word embedding models (Word2Vec, GloVe, FastText). The Urdu word vector sample was converted using neural networks or probabilistic models. It is very attractive as the learned vectors specifically convert the number of linguistic regularities and impressions into vectors. There is significance in the fact that many of these experiences can be expressed as linear translations. For example, the average of vector لاهور (Lahore), vector سرگودھا (Sargodha), and vector گوجرانوالہ (Gujjranwala) is the vector گوجرانوالا (Gujjranwala) that is similar to above words (Mikolov, 2013) (Tomas and Zweig, 2013). The explanation in the following sub-sections is about the details of word embedding models and overall model description.

3.1. Word Embedding Models

3.1.1. Word2Vec

In this study, Word2Vec a neural network model has been used (Mikolov, 2013) to convert Urdu words into vectors. It is composed of two sub-models (1) continuous skip-gram and (2) continuous bag of words (CBOW). These models can learn high-quality vector representations of words from a large amount of unstructured text. The skip-gram can predict contextual words based on a provided target word. It used smart weight adjustment and assigned lesser weights to those words which have more distance from the target word. In this study, we train and use the Word2Vec's skip-gram model as its structure of learning embeddings is similar to our Urdu WSD problem. The beauty of the skip-gram model lies in the fact that it requires less time complexity to embed words in the vector space.

This study used three Word2Vec models in the embedding layer one by one to the deep neural network model, two are pre-trained, and one custom-trained model. One was trained by Khurram (Kanwal *et al.,* 2019) with vector dimension size 300, and context window size 5 on the MK-PUCIT (926,776 tokens) corpus. The second was trained by Samar (Haider, 2018) by using Word2Vec's skip-gram model with 300 vector dimension size and 5 context window size on a combined Urdu (140M tokens) corpus. Third, to analyze the behavior of custom and pre-trained models, we trained Word2Vec's skip-gram model on the Ur-Mono (Jawaid *et al.,* 2014) corpus which has 95 million tokens, trained with 300 dimensions, and a context window of size 5.

3.1.2. GolVe

The second word embedding model used in this study was proposed by Jeffery and named GloVe (Pennington *et al.,* 2014) in 2014. It is a co-occurrence probabilistic model and can generate high-quality embeddings of words from global corpus statistics directly. As GloVe is a probabilistic model and word2vec is a neural network model, it is interesting to analyze the performance of these models on the Urdu WSD task in the embedding layer. However, it takes less training time as compared to the Word2Vec model. The GloVe uses a statistical approach for weight adjustment that is the least square. This study also analyzes whether GolVe can discriminate between relevant words سکول (School) and استاد (Ustad) from irrelevant words پانی (Pani) and پنکھا (Pankha) (Pennington *et al.,* 2014). Though the Word2Vec also discriminates against this type of words, GloVe performs better than the other models (Pennington *et al.,* 2014) on the number of tasks i.e., word similarity, word analogy, and named entity recognition.

In the experiments conducted as part of our study, one custom-trained GloVe model was implemented as an embedding layer of the deep neural network model to analyze its working in the Urdu WSD task. This GloVe model was trained on the Ur-Mono (Jawaid *et al.,* 2014) corpus with a dimension size of 300 and context window of size 5.

3.1.3. FastText

The third word embedding model used in the experiments conducted as part of our study was the FastText library, released by Facebook in 2017 (Bojanowski, 2017). It is an extension of (Tomas and Zweig, 2013)'s skip-gram model, also based on a neural network, used for creating word embedding from unstructured text. The beauty of FastText is to take the morphology of words into account and creates embeddings for out-of-vocabulary (OOV) words by converting words into the bag of words n-gram. The famous models Word2Vec and GloVe are not able to handle words that are out of their vocabulary (Bojanowski, 2017). For example, the word ڈیسکول (descol) is OOV for Word2Vec and GloVe but FastText creates an embedding for it due to its sub-word morphology.

Moreover, this study used three FastText models in the embedding layer, trained on different corpora. First, the pre-trained model by Facebook[1], trained on Common Crawl with vector dimension size 300, and a context window of size 5. Second, pre-trained by Khurram (Kanwal *et al.,* 2019) with a vector dimension size of 300, and a context window of size 5. Third, the model was custom trained with the same parameters and corpus as used for custom-trained Word2Vec and GloVe.

3.2. Model Description

Figure 1 describes the architecture of the deep neural network model for the Urdu WSD task. It consists of four layers including two hidden layers. The first layer is named as the embedding layer where text with labels is given as input, it converts text into embeddings which are semantically rich vectors with low dimensions (Fang *et al.,* 2016).

As discussed in Section 3.1 this study used seven embedding models, one by one, to generate these embeddings. The two hidden layers were implemented on top of the embedding layer by using the Keas library. These hidden layers used five deep learning models including simple RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU one after another.

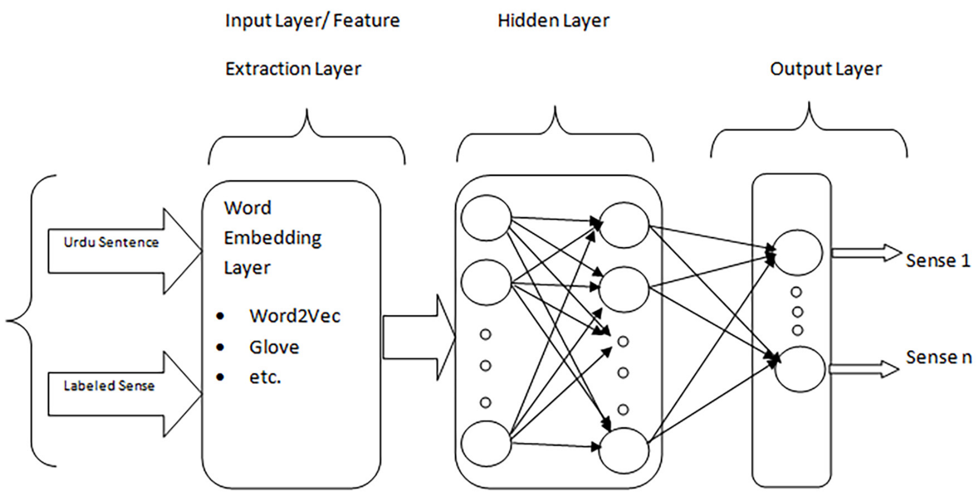


Figure 1. Deep learning model architecture based on pre-trained and custom trained word embedding model layer

[1] https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md

The parameters that the deep neural network used for the WSD task are shown in Table 2. Three different hidden layers—numbered one to three—were used in the experiments, each of which had 100 fixed neurons, Tanh as its activation function, a constant learning rate of 0.001 applied, a batch size of 32, a validation split of 20%, and two specific Softmax-activated neurons in the output layer.

This study employed an ideal number of epochs, wherein a large number of epochs can result in an over fitting issue while a little number could cause the model to be under-fit (Nisha, 2020). Epochs from 5 to 40 have been used in experiments, but 20 epochs were shown to be the best choice.

Table 2. Parametric configuration for all the experiments

Parameter	**Value**
Learning Rate	Default
Activation Function in Output Layer	SoftMax
Batch Size	32
Number of Epochs	20
Number of Neurons in Output Layer	2
Embedding	1000
Number of Neurons in Hidden Layer	100
Activation Function in Hidden Layers	Tanh
Validation Split	0.2
Dropout Value	0.2

4. Experimental Setup

This section describes the dataset, evaluation methodology, and evaluation measures used for the Urdu Word Sense Disambiguation (WSD) task.

4.1. Dataset

This study used (1) Urdu All Words-Word Sense Disambiguation (UAW-WSD-18) (Ali Saeed, 2019) and (2) Urdu Lexical Sample-Word Sense Disambiguation (ULS-WSD-18) corpora (Ali *et al.,* 2019).

First, UAW-WSD-18 (Ali Saeed, 2019) corpus consists of 5,042 words, 856 tagged instances, and 466 types. It was released by (Ali Saeed, 2019) in 2018 and is freely available. Sense ambiguity for each word was obtained from the Urdu Lughat (Ali *et al.,* 2019) dictionary, and in UAW-WSD-18 sense varied from two to eleven types. The class-wise statistics of UAW-WSD-18 are given in Figure 2.

Second, the ULS-WSD-18 (Ali *et al.,* 2019) corpus consists of 7,185 sentences, 222,533 tokens, 57,150 nouns, 25,719 verbs, 15,297 adjectives, 3,557 adverbs, and 120,810 other parts of speech (PoS) categories. The dataset was divided into train and test datasets, with 66% of the sentences (4,790) being used for training and 33% (2,395) being used for testing. This corpus contains each word with two to eight possible senses as explained in Figure 3.

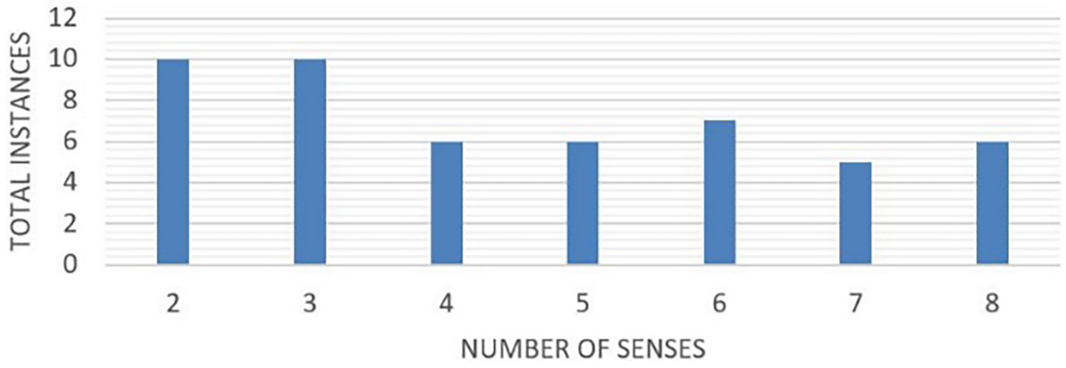


Figure 2. Sense wise detail of UAW-WSD-18 corpus

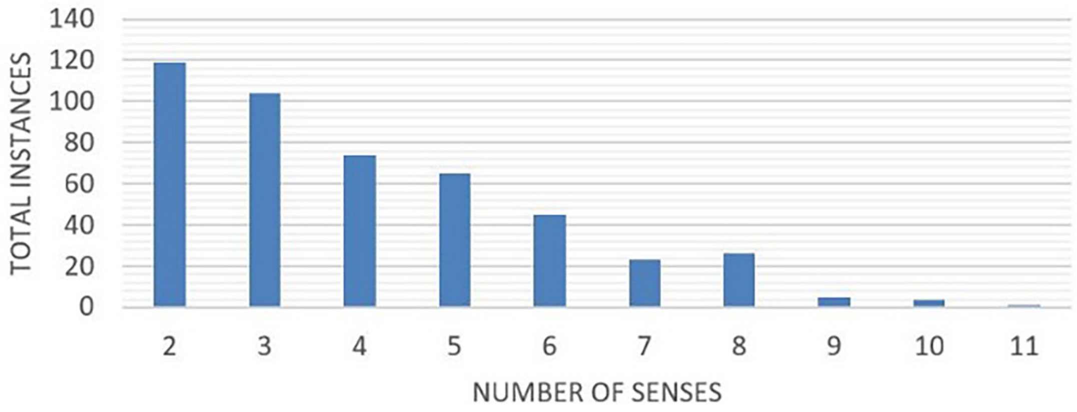
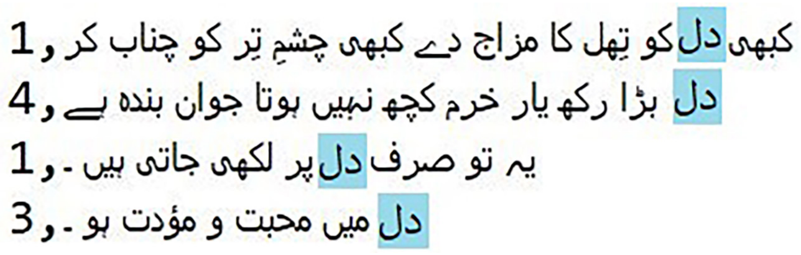


Figure 3. Sense wise detail of ULS-WSD-18

In Figure 4, an ambiguous word دل (Dil) is explained with its three senses. In the first and third instances of Figure 4, the word دل (Dil) is used as in sense 1, and in the second and fourth instances, it is used in its 4th and 3rd sense. However, دل (Dil) has five senses in total in the ULS-WSD-18 corpus.



کبھی دلکو تِھل کا مزاج دے کبھی چشمِ تِر کو چناب کر ,1

دل بڑا رکھ یار خرم کچھ نہیں ہوتا جوان بندہ ہے ,4

یہ تو صرف دلپر لکھی جاتی ہیں ۔ ,1

دل میں محبت و مؤدت ہو ۔ ,3

Figure 4. Instances from the ULS-WSD-18 corpus for دل (Dil) word

Deep learning models require datasets of a very high quality, however, the datasets used in this study have a sufficient number of quality instances for both Urdu WSD tasks (Saeed and Ali, 2021). Saeed and Ali (2021) applied deep learning models with Keras embedding layer. In the current study, we applied deep learning models with Urdu pre-trained and custom-trained word embedding layers to analyze the models' behavior. Table 3 compares the word embedding layer models used in the embedding layer of each study related to WSD in different languages. From Table 3 we analyze that most researchers used other than word embedding models in the embedding layer. It is a challenging task to train word embedding models and then use them in the embedding layer.

Table 3. Comparison of different deep learning models using different embedding layers (TS is an abbreviation of this study)

Sr. No	Deep Learning Model(s)	Dataset	NLP Task	Embedding Layer	Language	Ref
1	RNN, LSTM, GRU, Bi-LSTM, & ELM	ULS-WSD-18 Corpus	LS-WSD	Keras	Urdu	(Saeed and Ali 2021)
2	LSTM	Custom Dataset	LS-WSD	One-hot	Punjabi	(Varinder pal Singh 2020)
3	RNN, LSTM, GRU, Bi-LSTM, & ELM	UAW-WSD-18 Corpus	WSD	Keras	Urdu	(Saeed and Ali 2021)
4	Bi-LSTM	Japanese Sense Dataset	AW-WSD	nwjec2vec	Japanese	(Cao 2019)
5	RNN, LSTM, GRU & Bi-LSTM	ANERcorp Corpus	NER	Bag of word & AraVec 2.0	Arabic	(M. N. Ali 2018)
6	RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU	ULS-WSD-18 Corpus	LS-WSD	Word2Vec, GloVe, FastText	Urdu	TS
7	RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU	UAW-WSD-18 Corpus	AW-WSD	Word2Vec, GloVe, FastText	Urdu	TS

4.2. Evaluation Methodology

In the current study of Urdu WSD, both Lexical Sample and All-Words corpora have been evaluated by using deep neural network models, from which features were extracted by using word embedding models. In the deep learning models, the embedding layer is implemented with a number of word embedding models, including pre-trained Word2Vec (Samar and Khurram), FastText (Facebook and Khurram), and custom (Word2Vec, GloVe, and FastText) models trained on Ur-Mono corpus. Famous deep learning models such as simple RNN, LSTM, bidirectional LSTM, GRU, and bidirectional GRU are employed in the second layer. Section 3 provides explanations of word embedding models in detail. Overall, 35 tests were conducted, and Section 5 explains the outcomes.

4.3. Evaluation Measures

Accuracy, Precision, Recall, and F_1 are commonly used evaluation measures for text classification tasks of NLP, including WSD (Navigli, 2009) (Sokolova and Lapalme, 2009). The measurements obtained in this study are calculated using the following equations:

The accuracy of a system is defined as the total number of correct predictions.

$$Accuracy = \frac{Correct\ Cases}{All\ cases} \times 100 \tag{3}$$

The precision of a system is the ability of the classifier not to call a negative sample positive.

$$Precision = \frac{True\_Positive}{True\_Positive + False\_Positive} \tag{4}$$

The recall of a system is the ability of a classifier to find all the positive samples.

$$Recall = \frac{True\_Positive}{True\_Positive + False\_Negative} \tag{5}$$

F_1 measure is a specific relationship (harmonic mean) between precision (Pre) and recall (Rec).

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

5. Results and Analysis

Table 4 contains the accuracy, precision, recall, and F_1-measure results[2] for the experiment on the UAW-WSD-2018 corpus. In this study two approaches (pre-trained, custom-trained) to word embeddings have been used in the embedding layer for All[3] deep learning models. In Table 4, the row in bold shows the best result.

Table 4. Results obtained by All models (sample RNN, LSTM, GRU, Bi-LSTM, Bi-GRU) based on different word embedding (Word2Vec, GloVe, FastText) layers on the UAW-WSD-2018 corpus

Approach	Word Embedding Models	Model	Accuracy	Precision	Recall	F1-measure
Pre-trained	Word2Vec (Samar)	All	58.30	0.34	0.58	0.34
	Word2Vec (Khurram)	All	56.24	0.33	0.57	0.42
	FastText (Khurram)	All	54.77	0.30	0.55	0.39
	FastText (Facebook)	**All**	**60.07**	**0.36**	**0.60**	**0.45**
Custom trained	Word2Vec	All	59.72	0.36	0.60	0.45
	GloVe	All	57.24	0.33	0.57	0.42
	FastText	All	56.89	0.32	0.57	0.41

[2] Detailed results on UAW-WSD-18 and ULS-WSD-18 corpus can be downloaded from this https://comsatsnlpgroup.wordpress.com/

[3] The deep learning models including Simple RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU

The highest accuracy 60.07 is achieved by All deep learning models when FastText Facebook (Bojanowski, 2017) embeddings are used in the embedding layer. At second best, All deep learning models achieved an accuracy of 59.72 when using the custom-trained Word2Vec embedding layer. Moreover, from pre-trained models, word2vec trained by Samar got an accuracy of 58.30 on All deep learning models. The worst accuracy of 54.77 was achieved by Khurram FastText (Kanwal *et al.,* 2019) on all deep learning models. The possible reason behind the highest and worst accuracy is the size of the data sets on which these word embedding models have been trained, as almost all word embedding models have been trained on the same parameters but on different sizes and domains of datasets. Overall, pre-trained and custom-trained embedding models perform comparably better than the baseline Keras embedding layer. Moreover, the results illustrate that training word embedding efficiently and using it in a deep learning model as an input layer is a hard task.

It is also illustrated from the results that the best F_1- measure 0.45 got from pre-trained FastText embedding (Bojanowski, 2017) and custom-trained Word2Vec embeddings for All deep learning models. At second F_1-measure 0.42 is achieved by Khurram Word2Vec (Kanwal *et al.,* 2019) and custom GloVe embedding layer using All deep learning models. In F1-measure custom FastText got third position by achieving score 0.41 in embedding layer using All deep learning models. Worst F1-measure 0.34 achieved by Samar Word2Vec on All deep learning models. All deep learning models, including simple RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU, firstly depend on the embeddings used in the input layer and on other parameters.

The results analysis indicates that pre-trained word embedding models perform better than custom trained word embedding models used in the embedding layer for a number of deep learning models for Urdu All-Words WSD task.

Table 5 contains the results of the ULS-WSD-2018 corpus including accuracy, precision, recall and F1-measure results. In this study, two approaches (pre-trained, custom trained) to word embeddings have been used in the embedding layer for All deep learning models.

Table 5. Results obtained by the All (Simple RNN, LSTM, GRU, Bi-LSTM, Bi-GRU) model based on different word embedding (Word2Vec, GloVe, FastText) layer on ULS-WSD-2018 corpus

Approach	**Word Embedding Models**	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-measure**
Pre-trained	Word2Vec (Samar)	LSTM	70.82	0.54	0.71	0.60
	Word2Vec (Khurram)	GRU	70.92	0.54	0.71	0.60
	FastText (Khurram)	LSTM	70.92	0.54	0.71	0.60
	FastText (Facebook)	LSTM	70.80	0.54	0.71	0.60
Custom trained	Word2Vec	GRU	70.87	0.54	0.71	0.60
	GloVe	**Bi-GRU**	**70.93**	**0.54**	**0.71**	**0.60**
	FastText	LSTM	70.92	0.53	0.71	0.60

Clustered chart between Y_test and Y_predict for all 7 senses and 265 testing instances for All-Words WSD task is given in Figure 5. The highest accuracy 70.93 on ULS-WSD-2018 has been achieved by Bi-GRU on the custom-trained GloVe word embedding model layer. In this study, it did not perform well as its strength is discrimination, not relevance. Second, the number of word embedding models got an accuracy of 70.92 using a number of deep learning models, including Khurram Word2Vec using GRU, Khurram FastText using LSTM, and custom FastText using LSTM. Third, custom Word2Vec using GRU got an accuracy of 70.87. The worst accuracy of 70.80 was achieved by FastText Facebook using LSTM. On the other hand, an F_1-measure of 0.60 was obtained by almost every deep learning model on every Word2Vec embedding layer.

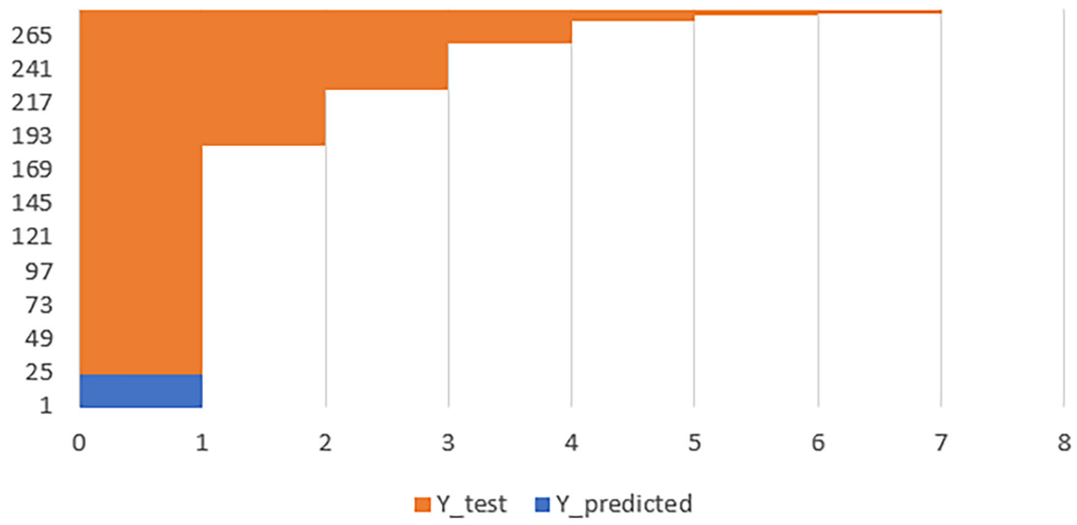


Figure 5. Clustered chart between Y_test and Y_predict for all 7 senses and 265 testing instances

6. Conclusion

This article compared various word embedding based methods for both Lexical Sample and All-Words WSD tasks for Urdu, a widely spoken language that is critically under-resourced in NLP research. Mainly this study identified that the most successful method for the All-Words Urdu WSD task is pre-trained FastText Word Embedding. This method exhibits top results (Accuracy = 60.07 and F_1 = 0.45). An additional contribution is the custom-trained Urdu WE models (Word2Vec, GloVe, FastText trained on Ur-Mono corpus) are publically available[4] to expand NLP research for Urdu language text. We intend to use BERT-based language models in the future to approach the problem of Urdu WSD.

[4] Word2Vec, GloVe and FastText models trained on Ur-Mono corpus are available at https://comsatsnlpgroup.wordpress.com/

References

Abid, M., A. H., Jawad, A., and Abdul, S., 2018. Urdu word sense disambiguation using machine learning approach. *Cluster Computing 21*(1), 515-522. https://doi.org/10.1007/s10586-017-0918-0

Ali, M., N., and Tan, G., and Hussain, A., 2018. Bidirectional recurrent neural network approach for Arabic named entity recognition. *Future Internet. 10*(12), 123. https://doi.org/10.3390/fi10120123

Ali, S., Nawab, R. M. A., Mark, S., and Paul, R., 2019. A word sense disambiguation corpus for Urdu. Language Resources and Evaluation. 53: 397-418. https://doi.org/10.1007/s10579-018-9438-7

Ali, S., Rao, M. A. N., Mark, S., and Paul, R., 2019. A Sense Annotated Corpus for All-Words Urdu Word Sense Disambiguation. *ACM Transactions on Asian and Low-Resource Language Information Processing, 18*(4), 1-14. https://doi.org/10.1145/3314940

Archana, K., and DK, L., 2020. Word2vec's Distributed Word Representation for Hindi Word Sense Disambiguation. In *International Conference on Distributed Computing and Internet Technology*, 325-335. https://doi.org/10.1007/978-3-030-36987-3_21

Arif, S. Z., Muhammad, M. Y., Atif, R., Fuzel, J., and Jamil F., 2016. Word sense disambiguation for Urdu text by machine learning. *International Journal of Computer Science and Information Security 14*(5).

Bojanowski, P. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics, 5*, 135-146. https://doi.org/10.1162/tacl_a_00051

Broda, B., Pawe, K., Micha, M., Adam, R., Radosaw, R., & Wardyski, A., 2013. Fextor: A feature extraction framework for natural language processing: A case study in word sense disambiguation, relation recognition and anaphora resolution. *Computational Linguistics, 458*, 41-62. https://doi.org/10.1007/978-3-642-34399-5_3

Cao, R., Bai, J., & Shinnou, H., 2019. Semi-supervised learning for all-words WSD using self-learning and fine-tuning. In *Proceding of 33^{r} Pacific Asia Conference Language, Information Computing*, 356-361.

Cotton, P., E., & Scott., 2001. SENSEVAL-2: Overview. In *Proceedings of the 2nd International Workshop on Evaluating Word Sense Disambiguation Systems*, 1-5.

Das, D., Debapratim, A. K., Soharab, H. S., & Rajat, K. P., 2023. A dataset for evaluating Bengali word sense disambiguation techniques. *Journal of Ambient Intelligence and Humanized Computing*, 1-30.

Dongsuk, K. O., Kim, S., Ko, K., & Youngjoong, 2018. Word sense disambiguation based on word similarity calculation using word vector representation from a knowledge-based graph. In the Proceedings of the 27th international conference on computational linguistics.

Fang, W., Jianwen, Z., Dilin, W., Zheng, C., & Ming, Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In the Proceedings of the 20th SIGNLL conference on computational natural language learning, 260-269. https://doi.org/10.18653/v1/K16-1026

Haider, S. 2018. Urdu Word Embeddings. In the Proceedings of the Eleventh International Conference on Language Resources and Evaluation.

Hussain, A. N., & Sarmad, H., 2009. Supervised Word Sense Disambiguation for Urdu Using Bayesian Classification. Center for Research in Urdu Language Processing, Lahore, Pakistan.

lgen, B., Eref, A., & Cneyd, A. T. 2012. The impact of collocational features in Turkish word sense disambiguation. In the Proceeding of 2012 *IEEE 16th International Conference on Intelligent Engineering Systems (INES),* 527-530. https://doi.org/10.1109/INES.2012.6249891

Kanwal, S., Kamran, M., Khurram, S., Faisal, A., & Zubair, N., 2019. Urdu Named Entity Recognition: Corpus Generation and Deep Learning Application. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 19*, 1-13. https://doi.org/10.1145/3329710

Kashif, R., 2010. Rule-based named entity recognition in Urdu. In Proceedings of the 2010 Named Entities Workshop.

Khan, W., Ali, D., Jamal, A. N., & Tehmina, A., 2016. A survey on the state-of-the-art machine learning models in the context of NLP. Kuwait Journal of Science 43(4): 66--84.

Le, M., Marten, P., Jacopo, U., & Piek, V., 2018. A deep dive into word sense disambiguation with LSTM. In the Proceedings of the 27th international conference on computational linguistics.

Jawaid, B., Kamran, A., & Bojar, O. 2014. A Tagged Corpus and a Tagger for Urdu. In LREC, Vol. 2, pp. 2938-2943.

Mihalcea, R., Timothy, C., & Adam, K., 2004. The SENSEVAL-3 English lexical sample task. In the Proceedings of SENSEVAL-3, the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text.

Mikolov, T. 2013. Efficient estimation of word representations in vector space. In the Proceeding of Internation Conference on Learning Representations.

Mir, T. A, Lawaye, A. A., Rana, P., & Ahmed. G., 2023. Building Kashmiri Sense Annotated Corpus and its Usage in Supervised Word Sense Disambiguation. *Indian Journal of Science and Technology, 16*(13), 1021-1029. https://doi.org/10.17485/IJST/v16i13.2396

Navigli, R., 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR), 41*(2), 1-69. https://doi.org/10.1145/1459352.1459355

Nisha, K., 2020. Sentiment Analysis of Regional Languages Written in Roman Script on Social Media. In *Data Science and Intelligent Applications*, 113-119. https://doi.org/10.1007/978-981-15-4474-3_13

Pennington, J., Richard, S., & Christopher, D. M., 2014. Glove: Global vectors for word representation. In the Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).

Rahman, T. 2004. Language policy and localization in Pakistan: Proposal for a paradigmatic shift. In Proceedings of the SCALLA Conference on Computational Linguistics.

Ramya, P, & B Karthik. 2023. Word Sense Disambiguation Based Sentiment Classification Using Linear Kernel Learning Scheme. *Intelligent Automation & Soft Computing*, 2379-2391. https://doi.org/10.32604/iasc.2023.026291

Rim, L., Aloulou, C., B., & Lamia, H. 2017. Word Sense Disambiguation of Arabic Language with Word Embeddings as Part of the Creation of a Historical Dictionary. In the Proceeding of International Workshop on Language Processing and Knowledge Management.

Saeed, A., Nawab, R. M. A., Stevenson, M., 2021. Investigating the Feasibility of Deep Learning Methods for Urdu Word Sense Disambiguation. *Transactions on Asian and Low-Resource Language Information Processing, 21*(2), 1-16. https://doi.org/10.1145/3477578

Sarmad, H., 2008. Resources for Urdu language processing. In the Proceedings of the 6th Workshop on Asian Language Resources.

Sokolova, M., & Guy, L., 2009. A systematic analysis of performance measures for classification tasks. *Information processing and management 45*(4), 427-437. https://doi.org/10.1016/j.ipm.2009.03.002

Taghipour, K., & Ng, H. T., 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In the *Proceedings of the 2015 conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 314-323. https://doi.org/10.3115/v1/N15-1035

Tomas, M., & Geoffrey, Z., 2013. Linguistic Regularities in Continuous Space Word Representations. In the Proceedings of NAACL-HLT.

Uslu, T., Alexander, M., Daniel, B., & Wahed, H., 2018. FastSense: An efficient word sense disambiguation classifier. In the Proceedings of the Eleventh International Conference on Language Resources and Evaluation.

Varinder, P. S. & Parteek, K., 2020. Word sense disambiguation for Punjabi language using deep learning. *Neural Computing and Applications. 32*(8). 2963-2973. https://doi.org/10.1007/s00521-019-04581-3

Wang, Y., Wang, M., & Hamido, F., 2020. Word sense disambiguation: A comprehensive knowledge exploitation framework. *Knowledge-Based Systems, 190*, 105030. https://doi.org/10.1016/j.knosys.2019.105030

Wu, Y., Jun, X., Yaoyun, Z., & Hua, X., 2015. Clinical abbreviation disambiguation using neural word embeddings. *Proceedings of BioNLP, 15*. https://doi.org/10.18653/v1/W15-3822

Zhang, X., Zhang, R., Xiaoyang, L., Fanshuang, K., Junfan, Ch., Samuel, M., & Yongyi, M. 2023. Word Sense Disambiguation by Refining Target Word Embedding. In the Proceedings of the ACM Web Conference. https://doi.org/10.1145/3543507.3583191