Saraiki Language Word Prediction And Spell Correction Framework

Muhammad Farjad Ali Raza, M. Asif Naeem
School of Computing,
National University of Computer and Emerging Sciences
Islamabad, Pakistan.
farjadmohal@gmail.com, asif.naeem@nu.edu.pk

***Abstract*—Word prediction, spelling error correction and finding similarity between words are very useful features in any language. The Saraiki is one of the popular languages spoken in Pakistan. To the best of our knowledge, very little work has been done in the literature for word prediction, spell correction and finding similar words for the Saraiki language. In this paper we address these issues by presenting a novel approach for word prediction, finding similar words, and spell correction in the Saraiki language. To achieve this, we used CBOW and Skip-Gram for the vectorization of the Saraiki language. From our results, we achieved word prediction accuracy of 24% in case of word2vec while 29% in case of the fastText. In case of word similarity, we achieved similarity score equal to 0.35, and 0.39 for word2vec CBOW and word2vec Skip-Gram respectively and similarity score of 0.35 and 0.41 for the fastText CBOW and the fastText Skip-Gram respectively. Our spell correction results show that as we increase wrong characters in words, the accuracy gets decreased. For sentence-level word prediction, we achieved accuracy of 63% in case of RoBERTa and 58% for distilled respectively.**

***Index Terms*—Word2vec, fastText, CBOW, Skip-Gram, RoBERTa, NLP**

I. Introduction

Saraiki is an Indo-Aryan language of the Lahnda group. There are 26 million speakers of the Saraiki language worldwide. It is mostly spoken in South Asian countries like Pakistan and India. According to 2017 census of Pakistan, there are 25.9 million people who speak the Saraiki, which is 12.19% of the total population of Pakistan. It is also known as the sweetest language of Pakistan. Every language has its own syntax, grammar and unique set of characters and words. Likewise, the Saraiki language has its own syntax, grammar, unique set of characters and words. Unique characters that are only found in the Saraiki are ٻ گ ڈ ڈ ٹ. Figure 1 shows an extensive Perso-Arabic alphabet script of all characters in the Saraiki language. Word prediction is a task of natural language processing, when writing text on a cell phone or querying something in Google search shows suggestions of the most likely words that will occur next. Word prediction can be found for many high resource languages like English.

[1]https://omniglot.com/writing/saraiki.htm

979-8-3503-9710-9/22/$31.00 ©2022 IEEE

Fig. 1. Perso-Arabic script of the Saraiki language[1]

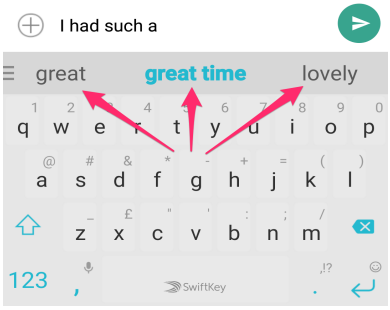

Fig. 2. Word predictions example[2]

Likewise, a model can be trained for the Saraiki language that can predict center word and next word. In Figure 2

[2]https://askubuntu.com/questions/1125541/word-prediction-like-android-keyboard

prediction of most likely words can be seen that come next when typing text on mobile phone screen. Like other languages, wrong spellings of words is also a problem in the Saraiki language. Statistical approaches use probability to predict correct word. In Figure 3 suggestions for the correct word can be seen when typing text on mobile phone screen.

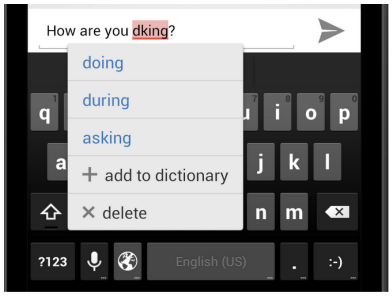

Fig. 3. Spell correction example[3]

Text of any language is first pre-processed and then sentences are separated, these sentences are then split into words. The process of separating sentences and splitting into words is called tokenization. To make those words meaningful to machines, words are represented as vectors, those vectors are called embeddings. One of the important tasks of NLP is to represent words with similar meanings near to each other using cosine similarity. The cosine similarity is used to measure the angle between two vectors, which may then be used to assess whether or not the vectors point in nearly the same direction. Two words can be different in syntactic structure but similar in meaning. Word2Vec and the fastText are two unsupervised models that can be trained to calculate similarity between words and predicting missing word using context words. Words can be represented in different dimensions of vectors and context sizes. For spelling errors correction, matrix-based calculation called Levenshtein Distance can be used to calculate the difference of characters between two strings. It is calculated based on number of operations like deletion, insertion and substitution. Transfer learning is popular technique for reusing gained knowledge of a model on other tasks. RoBERTa model is a replica of BERT, both use transformers for their learning, main task of transformer is to preserve context using another technique called attention. This attention method basically generates different vectors for same word in different context. RoBERTa takes sentence as input and tries to predict the missing word. Our contribution in this paper are following:

- **Generating corpus for the Saraiki language:** Text dataset is not available so we collected data from online sources.
- **Preprocessing raw dataset:** There is no tool for preprocessing text data of the Saraiki language we preprocessed dataset using combination of regex operations matching, searching and transforming are used along with urduhack library[4].
- **Finding similar words:** Words that are similar in meaning can be found with their numerical representation using cosine similarity.
- **Predicting missing word:** Word2vec and the fastText models are used to predict missing word at word level using context. RoBERTa is used to predict missing word at sentence-level.
- **Correcting spelling errors:** Spelling errors correction, is done by using levenshtein distance.
- **Performance evaluation:** We evaluated results of similarity, spell correction and missing word prediction. fastText performed well on similarity and word prediction task using word level embeddings. Increasing number of wrong characters decreased accuracy. Word prediction using RoBERTa at sentence-level performed better than word level models.

II. Related Work

Vector representations of words can capture the syntactic and semantic meaning of words, Mikolov et al. [1] proposed vector offset method to solve the word analogy problem and generate word embeddings using RNNs. In [2] Mikolov et al. proposed two architectures, one of which is a continuous bag of words (CBOW) and other is Skip-Gram. In [3] The Skip-Gram was used to reduce training time and increase the quality of vector representations by using negative sampling and subsampling technique. Urdu word embeddings are generated using the word2vec model the results were compared and word2vec outperformed LSTM, Bi-LSTM and Tree-LSTM [4]. Words in a window of certain training samples may impact similarity, and the word on the left and right should have more weight than other far words [5]. In [6] word2vec model is evaluated based on different window sizes and vector dimensions for semantic similarity tasks. In [7] deep learning-based feed-forward network is used to recognize the Saraiki language words. To build the Wordnet database, the lexico-semantic relationships of nouns are analyzed for the Saraiki language, which are primarily singular and plural [8]. A hierarchical POS tagger is proposed for the Saraiki language, but due to resource scarceness, most parts of speech were taken from urdu language [9]. Bilingual word embeddings are evaluated based on semantic and similarity based evaluations. Human scores tend to be more dissimilar as degree of similarity increases [10]. Urdu Wordnet corpus is used to map senses and POS tags, Urdu language is first translated to the Saraiki and then mapped with POS tags [11]. The fastText model is used to generate embeddings and those embeddings were evaluated on state-of-the-art test dataset wordsim-353 and simlex-999 [12]. Bert is trained from scratch in Roman Urdu and results were compared on monolingual, bilingual and multilingual BERT for masked language modelling and next sentence prediction task [13]. Masking percentage is increased

[3]https://android-developers.googleblog.com/2012/08/creating-your-own-spelling-checker.html

[4]https://urduhack.com/

up to 40% and results were improved [14]. Minimum edit distance or edit operation necessary to change string str1 to string str2, each matrix cell is populated with the difference between the edit operations completed [15]. Urdu embeddings was generated using word2vec Skip-Gram model. Model is evaluated using wordsim-353 and simlex999 test datasets, Spearman correlation is then calculated to measure the quality of embeddings [16]. A. Altowayan and L. Tao [17] suggested an Arabic word embedding approach to extract characteristics for opinion mining, proposed model creates and trains vectors of the Arabic corpus. RoBERTa pre-trained model was fine-tuned for the Dutch language and results were compared, it outperformed many BERT base models when dealing with smaller datasets [18]. An individual vector is generated for each word in the input sequence that is scanned by attention [19]. BERT [20] A deep learning-based model is used to create a bidirectional encoder representation using transformers, It gains an understanding of the connections among the words in a given input text. In [21] The results of using BERT to generate hashtags for tweets were compared to those of other word embedding models. Contextual embeddings of BERT are compared against context-free embeddings of the word2vec, fastText, and GloVe, the findings indicate that contextual embeddings are more accurate [22]. Akdemir et al. [23] compared results of the word2vec, fastText and mBERT on different NLP tasks, the dimensions are set to 768 in order to provide a fair comparison. mBERT performed better than any of the other embeddings. RoBERTa performed way better than BERT on many state-of-the-art datasets [24]. As per literature and to the best of our knowledge very little work has been done for the Saraiki language.

III. Proposed Solution

Figure 4 presents a detailed work-flow for our approach. The first step in our methodology is to collect the dataset, then it is processed and tokenized into sentences and words. For spell correction, tokenized words are used. Tokenized words are then provided as input to gensim library for training CBOW and Skip-Gram models. Quality of word embeddings is then evaluated. Same tokenized dataset of sentences is provided as input to RoBERTa for training. Results are then compared for word prediction accuracy.

1) Data Collection: Text Data is collected from different sources. The Saraiki news, the Saraiki wikipedia and Quran with the Saraiki translation. The collected text data is combined and saved into one file.

2) Data Preprocessing: Text data from online websites contain a lot of irrelevant information that need to be removed. Special characters, latin characters, digits, alphabets, empty spaces, emoticons, tab spaces, and new lines were removed. Compound words containing character vow (و) in the center when tokenizing were separated because of space in between e.g (اطلاعات و نشريات). Underscore before and after character _و_ solves the problem. Character (ھ) was replaced with

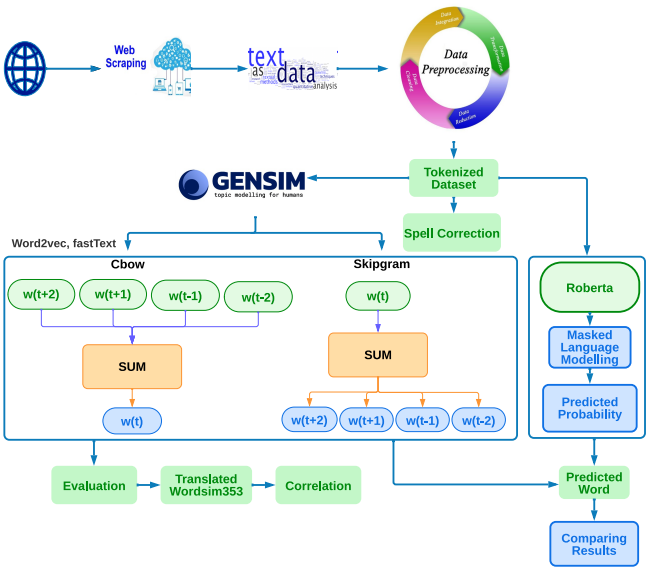

Fig. 4. Proposed Framework Diagram

(ہي) then (ۓ) with (اُ) and ء with ي. Cleaned dataset is prepared for further processing.

3) Data Preparation For Training: Text data after pre-processing is then passed to the popular natural language processing library, known as Spacy[5] to tokenize sentences and words from paragraphs.

4) Word Level Model Configuration: Gensim is very popular libarary for generating word embeddings. [6] Word2vec and fastText are pre-built modules in gensim to generate word emebeddings. We initialized the word2vec model with different vector sizes ranging from 50-400. The minimum count of each word is set to 1 and window size is set to 2-6. After that, vocabulary is built for each vector size by passing words to the model. Each model is trained for one hundred epochs.

5) Spell Correction Configurations: Hundred words are randomly selected to measure the accuracy of spell correction, those words are then incorrected by randomly replacing one to five characters at a random index of words.

6) Sentence-Level Model Configurations: Tokenized sentences are provided as input to the RoBERTa. Pre-trained model and tokenizer of the RoBERTa model are used. Tokenizer trained on new words with a batch size of 1,000 and vocabulary size is set to 103,345. Model token embedding size is then resized to the new length of tokenizer. There are 103,198 sentences for training and 25,800 for testing. After splitting, the dataset is then tokenized with newly trained tokenizer. The text is then grouped into block sizes of 128 for both train and test datasets. After grouping, there are 16,202 groups of training and 4,067 of testing. Masked language modelling is set to 15%.

[5]https://spacy.io/
[6]https://radimrehurek.com/gensim/

IV. Experiments

A. Setup

Dataset contains 129,145 sentences of the Saraiki language. Two hundered sentences with missing word at random place were used to measure accuracy of word prediction. Hundred words were used to measure the accuracy of spell correction. Word level models are trained with vector size 50 to 400 using Google colab environment. Word level model with vector size 768 and sentence-level models require a lot of memory. Kaggle provides free access to GPU and provides much bigger RAM. Word level model with vector size 768 is trained in Kaggle. Sentence-level model RoBERTa, default vector size is 768 and it is also trained in Kaggle.

B. Results

1) Semantic Similarity: Embeddings can be visualized in semantic space. Principle component analysis is applied to reduce the dimensionality of vectors. Cosine similarity is calculated between word vectors. Words with high cosine similarity appear close to each other or words with similar meaning appear close in semantic space.

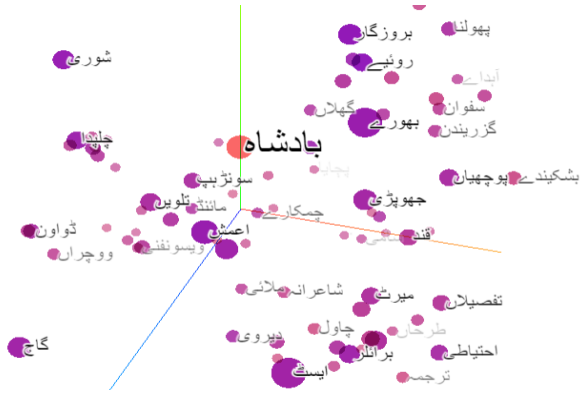

Fig. 5. Similar words in semantic space

2) Word Similarity: Word2vec similarity is measured using wordsim-353 that is translated into the Saraiki. Different window sizes and vector sizes are used to evaluate. From Table I it can be observed that highest similarity for Word2vec CBOW is 35% on window size 3 and vector size 100.

TABLE I
WORD2VEC (CBOW) SIMILARITY

Window Size	Wordsim-353					
	Vector Dimensions					
	50	100	200	300	400	768
2	0.303	0.311	0.320	0.304	0.304	0.297
3	0.350	**0.357**	0.315	0.325	0.271	0.326
4	0.292	0.264	0.285	0.274	0.272	0.323
5	0.311	0.285	0.259	0.266	0.276	0.310
6	0.258	0.238	0.255	0.274	0.314	0.307

From Table II it can be observed that highest similarity for word2vec Skip-Gram highest similarity is 39% on window size 6 and vector size 50.

TABLE II
WORD2VEC (SKIP-GRAM) SIMILARITY

Window Size	Wordsim-353					
	Vector Dimensions					
	50	100	200	300	400	768
2	0.321	0.350	0.336	0.325	0.322	0.298
3	0.339	0.372	0.344	0.325	0.288	0.306
4	0.337	0.320	0.317	0.297	0.289	0.316
5	0.357	0.324	0.356	0.327	0.311	0.313
6	**0.392**	0.342	0.337	0.330	0.336	0.313

The fastText model similarity and accuracy is measured using different window sizes and vector sizes. From Table III it can be observed that highest similarity for the fastText CBOW is 35% on window size 6 and vector size 400.

TABLE III
FASTTEXT (CBOW) SIMILARITY

Window Size	Wordsim-353					
	Vector Dimensions					
	50	100	200	300	400	768
2	0.315	0.328	0.322	0.328	0.336	0.334
3	0.313	0.315	0.342	0.313	0.315	0.338
4	0.339	0.305	0.326	0.333	0.326	0.333
5	0.315	0.308	0.310	0.320	0.332	0.344
6	0.302	0.325	0.350	0.332	**0.358**	0.344

From Table IV it can be observed that highest similarity for the fastText Skip-Gram is 41% on window size 3 and vector size 100.

TABLE IV
THE FASTTEXT MODEL (SKIP-GRAM) SIMILARITY

Window Size	Wordsim-353					
	Vector Dimensions					
	50	100	200	300	400	768
2	0.374	0.363	0.358	0.368	0.337	0.335
3	0.393	**0.417**	0.377	0.342	0.322	0.331
4	0.387	0.353	0.380	0.348	0.338	0.338
5	0.400	0.371	0.365	0.351	0.344	0.332
6	0.400	0.354	0.353	0.338	0.380	0.338

3) Word Prediction Accuracy: In this section missing word prediction accuracy is measured. From Table V it can be observed that highest accuracy for word2vec word prediction is 24% on window size 3 and vector size 400.

TABLE V
WORD2VEC ACCURACY

Window Size	Wordsim-353					
	Vector Dimensions					
	50	100	200	300	400	768
2	0.160	0.175	0.195	0.205	0.215	0.205
3	0.115	0.165	0.145	0.165	**0.245**	0.160
4	0.135	0.185	0.160	0.165	0.155	0.155
5	0.115	0.155	0.140	0.145	0.145	0.115
6	0.115	0.130	0.130	0.140	0.120	0.120

From Table VI it can be observed that highest accuracy for the fastText word prediction is 29% on window size 3 and vector size 400.

TABLE VI
FASTTEXT ACCURACY

Window Size	Wordsim-353					
	Vector Dimensions					
	50	100	200	300	400	768
2	0.095	0.110	0.165	0.175	0.185	0.185
3	0.085	0.100	0.115	0.210	**0.295**	0.160
4	0.205	0.230	0.245	0.250	0.265	0.140
5	0.140	0.235	0.250	0.260	0.240	0.135
6	0.150	0.200	0.245	0.245	0.115	0.120

4) Spell Correction Results: From Table VII it can be observed that from results, as we are increasing wrong characters, accuracy for spell correction is decreasing.

TABLE VII
LEVENSHTEIN DISTANCE SPELL CORRECTION

Word Error	L.D Accuracy
1	0.79
2	0.46
3	0.39
4	0.26
5	0.17

5) Sentence-Level Word Prediction: Huggingface library is open source platform, used for fine tuning RoBERTa model. RoBERTa training and validation loss can be seen in Figure 6 base model after five epochs started to underfit and in Figure 7 distilled model after three epochs started to underfit, as validation loss is increasing.

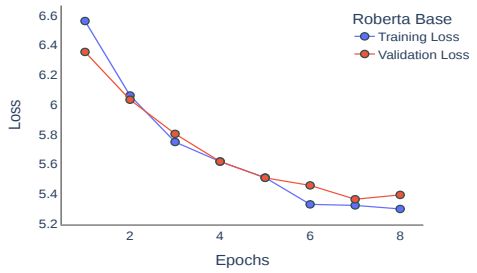

Fig. 6. RoBERTa base model training and validation

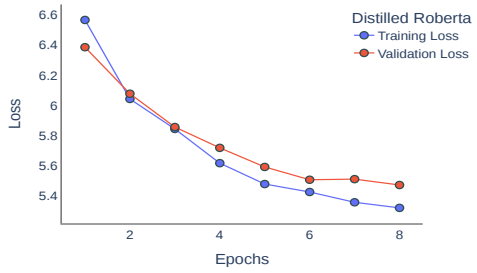

Fig. 7. RoBERTa distilled model training and validation

Two hundred sentences used to measure the accuracy of the Word2vec and the fastText are given as input to measure the accuracy of RoBERTa. Table VIII shows that the RoBERTa base model perplexity is low with high accuracy.

TABLE VIII
ROBERTA MODEL ACCURACY AND PERPLEXITY

Model	Perplexity	Accuracy
RoBERTa Base	217.29	63%
Distilled RoBERTa	234.12	58%

We took 10 sentences randomly, out of two hundred to visualize results for test purpose. Figure 8 shows masked words denoted with question mark in sentence column. In the target word column are the target words. Predicted column shows the predictions from our model. Score column shows the probability of predicted word.

Sentence	Target Word	Predicted	Score
بلوچستان دے صنعتی علاقے حب اچ واقع فیکٹری اچ گیس لیکیج نال دھماکے دے نتیجے اچ مزدور سڑ تے ؟ تھئ گئے	زخمی	زخمی	0.999
انہاں ایں گالھ دا اظہار اپنڑیں نال منتخب ھک پروگرام اپ کا وزیر اعظم اچ لوگاں نال ٹیلی فون ؟ گالھ مباڑ کریندے ہونے اکھیا	تے	تے	0.999
معیشت بہتر تھیندی پئی اے ٹیکس ریونیو ہزار ؟ ارب روپے تئیں پچ گئے وفاقی وزیر منصوبہ بندی اسد عمر	سو	اچ	0.050
بلوچستان دے صنعتی علاقے حب اچ واقع فیکٹری اچ گیس لیکیج نال دھماکے ؟ نتیجے اچ مزدور سڑ تے زخمی تھئ گئے	دے	دے	0.999
انہاں ایں گالھ دا اظہار اپنڑیں نال منتخب ھک پروگرام اپ کا وزیر اعظم اچ لوگاں نال ٹیلی فون تے گالھ مباڑ ؟ ہونے اکھیا	کریندے	کریندے	0.999
بلوچستان دے صنعتی علاقے حب اچ واقع فیکٹری اچ گیس لیکیج نال دھماکے دے نتیجے اچ مزدور سڑ تے زخمی ؟ گئے	تھئ	تھئ	0.999
انہاں ایں گالھ دا اظہار اپنڑیں نال منتخب ھک پروگرام اپ کا وزیر اعظم اچ لوگاں نال ٹیلی فون تے گالھ مباڑ کریندے ؟ اکھیا	ہونے	ہونے	0.997
معیشت بہتر تھیندی پئی اے ٹیکس ریونیو ہزار سو ارب روپے تئیں پچ گئے ؟ وزیر منصوبہ بندی اسد عمر	وفاقی	کہ	0.091
بلوچستان دے صنعتی علاقے حب اچ واقع فیکٹری اچ گیس ؟ نال دھماکے دے نتیجے اچ مزدور سڑ تے زخمی تھئ گئے	لیکیج	لیکیج	0.999
انہاں ایں گالھ دا اظہار اپنڑیں ؟ منتخب ھک پروگرام اپ کا وزیر اعظم اچ لوگاں نال ٹیلی فون تے گالھ مباڑ کریندے ہونے اکھیا	نال	نال	0.999

Fig. 8. RoBERTa model word prediction results

C. Discussion

Results of word2vec and fastText models are found to be more accurate on vector size 400 and window size 3 in the task of word prediction. Max correlation of word2vec CBOW is on window size 3 and vector size 100 and Skip-Gram on window size 6 and vector size 50. In fastext CBOW max correlation is on window size 6 and vector size 400 and Skip-Gram on window size 3 and vector size 100. Spell correction method works well on less character mistakes. RoBERTa base and distilled RoBERTa model accuracy on word prediction is compared and we found that RoBERTa base model is more

accurate because it has more parameters and layers. Perplexity score for RoBERTa base is lesser than the distilled RoBERTa. For fair comparison with RoBERTa we trained CBOW and Skip-Gram of both word2vec and the fastText with vector size 768 but the accuracy didn't improve.

V. Conclusion and Future Work

In our research, we generated text corpus for the Saraiki language. We improved the tokenization of words that use conjunction character و. We found that word level models are less accurate on predicting missing words. Increasing window sizes or the vector sizes did not improve results. Sentence-level model performed far better at predicting missing words. From spell correction results, we conclude that levenshtein distance accuracy decreased as we increased wrong characters in a word. In future, we plan to collect more data to improve the embeddings. Tokenizer plays an important role in separating sentences and words. We aim to develop separate word and sentence tokenizer for the Saraiki language. Part of speech tagging, optical character recognition and speech recognition are important problems. We also aim to solve these problems.

References

[1] T. Mikolov, W. tau Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Atlanta, Georgia), pp. 746–751, Association for Computational Linguistics, June 2013.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2013.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (Red Hook, NY, USA), p. 3111–3119, Curran Associates Inc., 2013.

[4] S. H. Kumhar, M. M. Kirmani, J. Sheetlani, and M. Hassan, "Word embedding generation for urdu language using word2vec model," *Materials Today: Proceedings*, 2021.

[5] C.-Y. Chang, S.-J. Lee, and C.-C. Lai, "Weighted word2vec based on the distance of words," in *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 2, pp. 563–568, 2017.

[6] D. Jatnika, M. A. Bijaksana, and A. A. Suryani, "Word2vec model analysis for semantic similarities in english words," *Procedia Computer Science*, vol. 157, pp. 160–167, 2019.

[7] M. Jan and Y. Saleem, "Optical character recognition (ocr) system for saraiki language using neural networks," *University of Engineering and Technology Taxila. Technical Journal*, vol. 21, no. 3, p. 106, 2016.

[8] N. Zamir, Benish, and J. Jahan, "Analyzing the lexico-semantic relationships of nouns used in the saraiki newspaper: A corpus based study," 2021.

[9] F. J. Saleemi, M. N. Asghar, S. Iqbal, M. U. Chaudhry, M. Yasir, S. U. Bazai, and M. Q. Khan, "A novel parts of speech (pos) tagset for morphological, syntactic and lexical annotations of saraiki language," *Journal of Applied and Emerging Sciences*, vol. 11, no. 1, 2021.

[10] J. P. Sanjanasri, V. K. Menon, S. Rajendran, K. P. Soman, and M. Anand Kumar, "Intrinsic evaluation for english–tamil bilingual word embeddings," in *Intelligent Systems, Technologies and Applications*, (Singapore), pp. 39–51, Springer Singapore, 2020.

[11] S. N. Sarah Gul, Musarrat Azher, "Development of saraiki wordnet by mapping of word senses: A corpus-based approach," *Linguistics and Literature Review*, vol. 7, no. 2, pp. 46–66, 2021.

[12] U. Khalid, A. Hussain, M. U. Arshad, W. Shahzad, and M. O. Beg, "Co-occurrences using fasttext embeddings for word similarity tasks in urdu," *CoRR*, vol. abs/2102.10957, 2021.

[13] U. Khalid, M. O. Beg, and M. U. Arshad, "RUBERT: A bilingual roman urdu BERT using cross lingual transfer learning," *CoRR*, vol. abs/2102.11278, 2021.

[14] A. Wettig, T. Gao, Z. Zhong, and D. Chen, "Should you mask 15% in masked language modeling?," 2022.

[15] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Commun. ACM*, vol. 7, p. 171–176, mar 1964.

[16] S. Haider, "Urdu word embeddings," in *LREC*, 2018.

[17] A. Altowayan and L. Tao, "Word embeddings for arabic sentiment analysis," pp. 3820–3825, 12 2016.

[18] P. Delobelle, T. Winters, and B. Berendt, "Robbert: a dutch roberta-based language model," 2020.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[21] M. Kaviani and H. Rahmani, "Emhash: Hashtag recommendation using neural network based on bert embedding," in *2020 6th International Conference on Web Research (ICWR)*, pp. 113–118, 2020.

[22] S. Deb and A. K. Chanda, "Comparative analysis of contextual and context-free embeddings in disaster prediction from twitter data," *Machine Learning with Applications*, vol. 7, p. 100253, 2022.

[23] A. Akdemir, T. Shibuya, and T. Güngör, "Subword contextual embeddings for languages with rich morphology," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 994–1001, 2020.

[24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.