A Survey of Word Embedding Algorithms for Textual Data Information Extraction

Eugen Vušak*, Vjeko Kužina*, and Alan Jović*
* University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10 000 Zagreb, Croatia
{eugen.vusak, vjeko.kuzina, alan.jovic}@fer.hr

***Abstract* - Unlike other popular data types, such as images, textual data cannot be easily converted into a numerical form that machine learning algorithms can process. Therefore, text must be embedded into a vector space using embedding algorithms. These algorithms attempt to encapsulate as much information as possible from the text into a resulting vector space. Natural language is complex and contains numerous layers of information. Information can be obtained from a sequence of characters or subword units that make up the word. It can also be derived from the context in which a word occurs. For this reason, a variety of word embedding algorithms have been developed over time, which use different pieces of information in different ways. In this paper, the currently available word embedding algorithms are described and it is shown what kind of information these algorithms use. After analyzing these algorithms, we discuss how it can be advantageous to use combinations of different types of information in different research and application areas.**

***Keywords* - word embedding, textual data, natural language processing, word space, text mining**

I. INTRODUCTION

Some might argue that natural language is the most complicated invention in human history. It is an ever-evolving system that allows us to express even the most elaborate ideas and is seemingly limitless. It is also one of the primary means of communication, as most knowledge is stored in some form of natural language. Natural language has formed over a very long period of time, resulting in a highly complex system that can represent large amounts of information in a very condensed form. This is often because some information can be inferred from prior knowledge and does not need to be said explicitly. Another reason is that closely related words have a similar internal structure and therefore the meaning of a word can be inferred even for unfamiliar words. This suggests that there are multiple layers of information in a given text, and knowledge of this information is crucial for designing an appropriate text mining algorithm.

In order for machine learning algorithms to work with textual data, the data must first be transformed into numerical data that is understandable to an algorithm. This is not as straightforward as with images, which can be directly represented by numbers. Textual data is usually transformed into vectors by word embedding algorithms. These are, by and large, models trained to perform a mapping of words or phrases into a real-valued vector of fixed size such that some desirable semantic properties and linguistic word relations are satisfied. Word embedding algorithms are de facto standard for improving model performance in numerous Natural Language Processing (NLP) tasks.

The aim of this paper is to describe the currently available word embedding algorithms and to highlight which information sources they use. For each algorithm, the focus is on transforming from a variable-length word or phrase to a fixed-dimensionality vector representation. The algorithm may use the entire available text to determine in which context a word or phrase occurs frequently, or subword units to obtain a representation. When subword units are the smallest possible, we speak of character-level information extraction. In this paper, we distinguish between word embeddings and contextualized word embeddings, where the latter can represent a word with different vector representations based on the context surrounding that word. While contextualized word embedding algorithms have proven themselves in most NLP applications with remarkable results, there is still room for (uncontextualized) word embeddings when no context is available, which may be the case when dealing with structured data or when computational resources are scarce.

The structure of this paper is as follows. Section II introduces and discusses related work. Sections III to V deal with information used to learn traditional word embeddings. Section III describes algorithms that use context to learn embeddings, while Section IV gives an overview of models that use additional information available in subword units. Section V gives an overview of algorithms that consider the word as a sequence of characters to represent it as a vector. Section VI gives a brief overview of some important recent advances in contextualized word embeddings. Finally, section VII draws concise conclusions about the topics covered in the paper.

II. RELATED WORK

Because of the vast amount of data stored as natural language and the implications that understanding it has for the study of human intelligence, the NLP field is active and well researched. Word embedding is one of the first steps in most NLP-related tasks and, accordingly, many papers have appeared that describe it systematically. In this paper, the problems faced by researchers, solutions to these problems and challenges of current methods are mentioned. In this section, an analytical evaluation of several recent scientific reviews dealing with word embeddings is provided.

In their study, Camacho-Collados and Pilehvar [1] have mainly described vector space models and word embeddings. Despite their flexibility and success in capturing the semantics of words, the embedding algorithms have significant limitations. These limitations stem from the fact that a word is usually represented by a single vector, which severely limits the ability of algorithms to generalize. They point out that words with multiple meanings may have a different definition depending on the context. For example, the word "cut" can have 70 different meanings and can even be interpreted as a noun, verb, or an adjective. The authors of the paper refer to this limitation as *meaning conflation deficiency*. They conclude that an accurate distinction of different senses is needed, which can be achieved by representing the different meaning of words, i.e., word senses as independent representations. The authors of the paper review several models that learn representations for such word senses.

In 2019, Kowsari et al. [2] provided a comprehensive survey of text classification algorithms to facilitate the selection of appropriate structures, architectures and techniques for the text classification task. Their survey covers a wide range of different methods that can be used for the said task, such as feature extraction and dimensionality reduction. Although the paper takes a more overarching approach, the authors of the study also briefly addressed word embeddings as it is an important step in text classification. For the purpose of this research paper, the authors focused on a clear description of the underlying mechanisms for three well-known deep learning methods. They also briefly discussed a novel approach to word representation, Contextualized Word Representations.

In the wake of public awareness of the digital language divide, Ruder et al. [3] authored a survey that provides an in-depth exploration of cross-lingual word embedding models. In this study, the authors described the benefits of such models and compare many models that have been proposed in recent years. Considering that many cross-lingual embedding algorithms have been derived from monolingual models, the authors first presented conventional monolingual models. As one of their most notable contributions, they also formulated a topology of cross-lingual models based on data requirements, chosen architecture, hyperparameters and additional fine-tuning.

The study by S. Wang et al. [4], provided an extensive and beneficial examination of word embedding architectures based on deep neural networks. To this end, the authors first identified and elaborated on classical word embedding models. They also noted some shortcomings that such models have. The first drawback addressed is the inability to represent words that were not present in the dataset used for training, also known as Out-of-Vocabulary (OOV) words. They presented a few state-of-the-art models that are able to represent such words with a vector derived from that word itself. Then, the authors focused on popular embedding models that are adapted to uniquely represent a word based on the context surrounding the same word. The last challenge addressed the processing of different languages with different linguistic structures. The paper mentioned models developed for the Chinese language, mainly because it is different from most European languages targeted by classical word embeddings. In addition, relevant neural network architectures commonly used in NLP were briefly identified and summarized.

Y. Wang et al. [5] provided a comprehensive overview of how word representations have evolved from static to dynamic. Static word representations are referred to as such because once learned, they do not change with context. As mentioned earlier, many words or phrases can be assigned different meanings that simpler models cannot represent. Authors call this *the polysemy problem.* On the other hand, dynamic word representations can represent the same word with different vectors based on the context, mitigating the polysemy problem. The paper described static embedding models and additionally presents some efforts to solve the polysemy problem. Next, the authors focused on recent dynamic word representation models and attempted to demonstrate how such models mitigate the said problem. They also provided insight into methods and datasets used for intrinsic and extrinsic evaluation of word embeddings. Finally, cross-lingual word embedding models were addressed, for both static and dynamic models.

III. CONTEXT-LEVEL LEARNED MODELS

Word embedding models that use only other words enclosing the selected word to learn word representations are called context-level learned models. Models of this type were developed on the basis of famously articulated Firth's 1957 notion "you shall know a word by the company it keeps." This means that semantically similar words or phrases often occur in a similar context. For example, suppose two very similar sentences. "A bee is buzzing around." and "A fly is buzzing around". We can conclude that bee and fly are closely related, which in a sense they are. This allows us to capture many semantic properties that a word may have, especially for more commonly used words. One of the most notable drawbacks of these models is their inability to effectively represent OOV words. Newer context-level learned models are computationally efficient and still desirable for this reason. They can also be extended with subword data.

A. NNLM

In 2001, Bengio et al. [6] published an innovative paper, in which they presented a novel idea of using neural networks to model the sequence as a joint conditional probability of the next word given all the previous words in a sequence, also known as neural network language model (NNLM). The language model can be expressed by:

$$p(w_1, w_2, \dots w_n) = \prod_{i=1}^{n} p(w_i | w_1, \dots w_{i-1}) \qquad (1)$$

The authors proposed to associate each unique word in a vocabulary with a dense real-valued vector of fixed size. The size of the feature vector representation was chosen to be much smaller than the size of the vocabulary, consequently alleviating the well-known curse of dimensionality. Feature vectors were computed for a given sequence and used to represent the joint probability function (1) with the neural network. Feature vectors and function parameters can, therefore, be learned

simultaneously. NNLM uses a multilayer neural network where the first layer maps a word from a sequence to its real-valued vector representation using an embedding C. These representations are concatenated and used to compute probability scores using a feedforward neural network with $tanh$ activation function and softmax layer to normalize the probabilities. Because the feature vectors are concatenated and fed into a feedforward network, the context length must be fixed, which prevents the network from using a longer context. Training of the neural network was accomplished by maximizing the log-likelihood on a training corpus. Optimization parameters are embedding C and network parameters θ. Although the authors focused on a specific model architecture, they pointed out that more complex architectures can be used in conjunction with the same embedding type.

B. SENNA

Following the work of Bengio et al., whose goal was to model the probability of a word given previous words in a sentence, Collobert et al. [7] proposed an architecture that instead uses a pairwise ranking criterion to compute scores describing the acceptability of a piece of text. Formally speaking, for a window $x = (w_1, w_2, \dots w_n)$ the $f(x|\theta)$ is the score with the network parameters θ. In this case, hinge loss is defined using ranking criterion as:

$$\sum_{x \text{ in} \mathcal{X}} \sum_{w \in \mathcal{D}} \{0, 1 - f(x|\theta) + f\left(x^{(w)}|\theta\right)\}, \qquad (2)$$

where $\mathcal{X}$ denotes the set of all possible text windows with n words, $\mathcal{D}$ is a dictionary of words and $x^{(w)}$ denotes a text window which is created by replacing the central word from x by the word w. This was done because a large dictionary size leads to a very complex computation of the normalization term in the softmax layer and would require more elaborate approximations. The architecture used in this work was named "SENNA" (Semantic/syntactic Extraction using a Neural Network Architecture) and is similar to NNLM, the main difference being the previously mentioned scoring. Some other notable differences are: (1) the removal of skip connections, (2) the use of $hardTanh$ instead of $tanh$. The paper also introduced a variable window that encompasses the whole sentence and then retrieves a fixed representation by using convolutional layer with max-over-time pooling.

C. Word2vec

Given the extended training times of previous work, Mikolov et al. [8] introduced Word2Vec, a novel word embedding architecture that extends the work of Bengio et al. by removing nonlinearity from NNLM to reduce computational complexity. They also adopt hierarchical softmax to reduce the problem of large number of classes. Hierarchical softmax represents the vocabulary as a Huffman binary tree and assigns short binary codes to frequent words, which further increases the training speed. In this paper, two different approaches for training word embeddings are proposed. The first is called Continuous Bag-of-Words (CBOW) and is similar to NNLM, where the non-linear layer is excluded and the projection layer, that maps the word indices of a context to a vector space, is shared across all words. The second architecture, Continuous Skip-gram or Skip-gram for short, is similar to CBOW, but instead of predicting a current word based on context, it attempts to predict surrounding words based on a current word.

In a follow-up paper, Mikolov et al. [9] provide additional optimizations to the word2vec algorithm. One of these optimizations is the implementation of the Negative Sampling method that is used to reduce the number of negative samples. This is done by selecting only random k negative samples instead of using them all.

D. GloVe

Pennington et al. [10] introduced the Global Vector or GloVe model to address some limitations that algorithms like word2vec have. The authors pointed out that such algorithms can only learn semantic similarities based on a limited local context window but are unable to use all the global statistical information available in the dataset. For this reason, GloVe uses a global word-word co-occurrence matrix to make efficient use of statistics. If we annotate this matrix with X, then each element X_{ij} represents the number of times the word w_j occurred in the context of a word w_i. The cost function is then defined as:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \widetilde{w_j} + b_i + \widetilde{b_j} - logX_{ij}\right)^2, \qquad (3)$$

where V denotes the size of the vocabulary and f is a weighting function used to weigh down rare co-occurrences, because they are usually noisy and carry less information.

IV. SUBWORD-LEVEL LEARNED MODELS

Despite their popularity and success, context-level learned models have some weaknesses. They model words independently and disregard any internal morphological structure, resulting in their inability to represent rare or morphologically complex words. To counter this, models have been developed that use subword information, either alone or in conjunction with context-level information. This approach is closer to how words are formed in natural language. For example, consider the terms "breakable" and "biased" and assume that we know their meaning. If we then learn that "unbreakable" is exactly the opposite of "breakable" we can conclude with enough data that the prefix "un" negates adjectives. Therefore, even if we have never heard the term "unbiased," we can infer that its meaning is the opposite of the term "biased". By using subword-level information, these models can significantly reduce problems with the words that are rare and OOV. For this reason, subword-level learned models are appropriate for tasks where morphological word structure contains fair amount of information. They are also practical for applications where there is no context available or plenty words are expected to be absent from vocabulary.

A. MorphoRNN

To better exploit the complex internal structures of English, Luong et al. [11] proposed a new model of architecture that compiles word representations using morphemes of words, called Morphological RNN (morphoRNN). For this purpose, the authors used an unsupervised morphological segmentation toolkit, called "Morfessor", which recursively splits words using hidden Markov models and labels morphemes with the tags pre (prefix), stm (stems) and suf (suffixes). For efficient learning, the input form of words is assumed to be $pre^* \; stem \; suf^*$. After the morphemes are retrieved, they are encoded by a simple embedding matrix described by Collobert et al. The encoded representations of the morphemes are then recurrently joined into a parent word p. For a pair: stem vector x_{stem} and affix vector x_{affix}, p is constructed as follows:

$$p = f\left(W_m\left(\left[x_{stem}; x_{affix}\right] + b_m\right)\right). \tag{4}$$

This forms the basis of the context-insensitive Morphological RNN (cimRNN), but to improve performance, the authors present a context-sensitive version called csmRNN that adopts the training approach proposed by Collobert et al. by adding a neural language model with a pairwise ranking criterion.

B. Byte Pair Encoding

Byte Pair Encoding (BPE) is not a word embedding algorithm, but a general-purpose data compression algorithm described by Gage in 1994 [12]. The BPE algorithm iteratively replaces all instances of most frequent pairs of adjacent bytes with a byte that was not in the original data. This is done until there are no more unused bytes or no more frequently occurring pairs.

In their 2016 paper, Sennrich et al. [13] focus on translating rare words using subword units due to the advantages described earlier. Their goal was to achieve an open vocabulary by using BPE with characters instead of bytes. Their algorithm works similarly to the original, but instead of bytes, the most frequent pairs of characters are merged to form a new character and added to the vocabulary. Although strictly speaking their study does not focus on word embedding, they introduced the novel idea of using BPE to construct subword units.

Recently, Heinzerling and Strube [16] extended this idea by introducing Byte-Pair Embedding (BPEmb). BPEmd is basically BPE applied to text, using the resulting symbols in combination with the GloVe word embeddings.

C. FastText

FastText [14][15] was developed in 2017 at Facebook as a direct extension of word2vec's Skip-Gram model, but taking subword information into account. Instead of predefined morphological structures, fastText models the morphology of words by representing them with character *n*-grams. Additionally, special boundary symbols < and > are added at the beginning and end of words to distinguish prefixes and suffixes from other strings. In practice, multiple n-grams of words are calculated for different values of n. The word w is also included to learn representations for each word. The Skip-Gram model is used to learn vector representations for n-grams (and word), and then the final word embedding is computed as the sum of all vector representations for a word.

V. CHARACTER-LEVEL LEARNED MODELS

Since words are essentially sequences of characters, some algorithms use character-level information to represent words as vectors. There are a few reasons to do this. For example, languages like Chinese do not use the alphabet, they use a logographic system of characters. This means that words are made up from symbols instead of letters, and each symbol can have its own meaning. Another reason is to avoid handcrafting features such as affixes and allow the model to learn the complex morphological structure. Empirically, end-to-end models have usually shown better results and higher generalization ability. Moreover, the number of unique characters is much more limited than the number of unique words. Especially when dealing with digitally written text, where characters are selected from a finite set. This fact allows models to correctly represent OOV words. Character-learned models should be used when subword information is insufficient as they are normally more elaborate. They should also often be combined with learned context-level models that can "memorize" the word semantic.

A. CWE

Chen et al. [18] proposed a new model for joint learning of character and word embeddings, which they named Character-enhanced Word Embedding model (CWE). Their goal was to take advantage of both internal characters and external context to create a model that is better adapted to languages where characters contain rich internal information. Their study used the CBOW model to demonstrate the CWE framework. CWE mainly consists of two different vocabularies. The first one is a Chinese character set, denoted with C and the second one is a Chinese word vocabulary, denoted as W. Each character $c_i \in C$ is represented by a vector $\bar{c}_i$ and each word $w_i \in W$ is represented by a vector $\overline{w}_i$ The final representation of the word $\bar{x}_j$ is represented as a composition between a vector w_j and the average of the embeddings of the characters forming this word. The authors considered two options for composition, addition and concatenation, but empirically found that concatenation does not significantly outperform addition and is more time-consuming. For this reason, they opted for addition. Consequently, a final representation for $\bar{x}_j$ can be written as:

$$\bar{x}_j = \frac{1}{2}\left(\overline{w_j} + \frac{1}{N_j}\sum_{k=1}^{N_j} \overline{c_k}\right). \tag{5}$$

In their work, Chen et al. also proposed several methods that assign multiple vectors to a single character. These methods are position-based, cluster-based, nonparametric cluster-based, and position-cluster-based character embeddings.

B. *Word embeddings based on convolutional neural networks*

Convolutional neural networks (CNN) are a famously proven network architecture for computer vision because they can learn spatial features, eliminating the need for feature extraction steps. The same advantages can be applied to word representation tasks. We can view a word or phrase as a one-dimensional image. For example, when we see a date or an address, we immediately know what it is by just looking at it. The fact that CNNs are very good at learning internal structures can be extended to learning complicated morphological structures that words may have. Because of this fact, many CNN-based approaches have been implemented over the years.

Dos Santos and Zadrozny [17] proposed such a method in 2014. They presented a deep neural network for performing part-of-speech (POS) tagging using joint representations at context and character levels. Their approach, called CharWNN, can be seen as an extension of the SENNA model, since for a given sentence network, it also takes the fixed-sized window of words around the target word to score it. Instead of using only a simple matrix-based word embedding, CharWNN also uses a convolutional layer to capture character-level information. Each character in a word is first transformed into a character embedding using a complementary embedding matrix. Then, using one-dimensional convolution, a matrix-vector operation is applied to each successive window in a sequence. Finally, max polling is used for all character windows in the word to extract a fixed size feature vector. Word-level embeddings were pre-trained in an unsupervised manner using the word2vec model. The character-level and word-level embeddings are then concatenated to form a final word representation.

Kim et al. [19] used a convolutional neural network at the character-level whose output serves as input to the language model. However, unlike the previously described model, this model did not use word embeddings. Given the large vocabulary, the word embedding matrices have numerous parameters that must be learned. When the authors removed these, they obtained a much smaller model. As they pointed out, this may be desirable in applications where resources do not support large models (e.g., embedded and mobile devices). In their model, words were embedded in the following steps: (1) characters were embedded through a small embedding matrix, (2) a one-dimensional character-level convolutional layer was used to extract features, (3) max-over-time pooling was applied to retrieve fixed-size word representations, (4) the resulting features were passed through a highway layer that further captures complex interactions between features. The authors showed that step 4 is not necessary, but it slightly improves performance. These word representations are then fed into a Long Short-Term Memory (LSTM) recurrent neural network with a softmax output to obtain distributions over the next word. Training is done by minimizing the cross-entropy loss between these distributions over the next word and the actual next word.

An interesting approach was proposed by Rama and Çöltekin [20], which used an LSTM autoencoder to compute representations for word pronunciations. Autoencoders traditionally have an hourglass shape where the first half, the encoder, learns to represent the input as an information-dense vector, and the second part of the model, the decoder, mirrors the first and learns to reconstruct the original input data from this compressed vector. After the autoencoder model is trained, the resulting dense vector in the middle is called the latent representation. Instead of the traditional autoencoder architecture, the encoder here is an LSTM network that "rolls" the word or phrase into a latent space, and then the decoder LSTM network "unrolls" it back to the original word or phrase. Thus, the model can learn word representations on an unlabeled dataset with variable input. Because the latent space representation is much smaller than the original vocabulary size, the model is forced to learn a sophisticated internal word structure and remove noise. Authors used this model primarily to visualize dialect shifts, but the same model could be adopted for many conventional NLP tasks.

VI. CONTEXTUALIZED WORD EMBEDDINGS

In the previous sections, we have focused mainly on the information used to train a model, and assumed that inference uses only one word to compute its vector representation. As mentioned, this approach has some problems, most notably the inability to represent multiple possible meanings of the same word. Therefore, novel models, called contextualized word embeddings, have been proposed. These models can have representations change based on context and are therefore better suited for usual unstructured text if resources are ample.

A. *ELMo*

Embedding from Language Models (ELMo) was introduced in 2018 by Peters et al. [21] to capture complex properties and variations across linguistic contexts of word use. The first step of the ELMo model is to compute uncontextualized word embeddings, which is purely character-based. This is achieved by convolutional filters, followed by two highway layers and a linear projection. Using these representations, ELMo essentially defines two language models with biLSTM network, one that models the probability of a token for previous tokens, and one that models the probability of a token for future tokens.

B. *GPT*

Radford et al. [22] introduced Generative Pre-Training (GPT) model. Instead of a bidirectional language model, GPT uses a unidirectional language model. For feature extraction, instead of LSTM layers, the GPT model uses the Transformer, a novel architecture that has proven to be very powerful in many different NLP tasks. This allows GPT to store more structured memory for handling long-term dependencies in the text, resulting in a more general model. BPE is used to extract subword information.

C. BERT

Devlin et al. [23] proposed Bidirectional Encoder Representations for Transformers (BERT), which uses a bidirectional transformer architecture instead of one-way architecture like GPT. They also introduced two unsupervised tasks for pre-training BERT: (1) Masked LM (MLM), where words are randomly masked and the goal is to predict the original vocabulary ID of a masked word based only on its context, (2) Next Sentence Prediction, in which the goal is to predict whether A follows B or not, for sentences A and B. As the successor of GPT, BERT also uses subword information retrieved from BPE.

D. XLNet

Yang et al. [24] introduced XLNet, which uses the Transformer-XL architecture and can learn bidirectionally by maximizing the likelihood over all permutations of the factorization order. It also avoids problems BERT caused by adding artificial symbols such as [MASK], resulting in XLNet outperforming BERT on 20 tasks. XLNet was pre-trained on subword pieces of known corpora.

VII. Conclusion

This paper has given an overview of current word embedding algorithms and presents a taxonomy based on the information used to compute word representation. We have shown the advantages of these sources and how they are used to solve some NLP challenges. Research in Natural Language Processing has recently gained momentum, and we have shown that significant progress has been made in word embedding methods in the last few years. Our analysis may allow future researchers to select or develop algorithms that are better suited for this task.

Acknowledgment

This work has been carried out within the project "Digital platform for ensuring data privacy and prevention of malicious manipulation of the personal data – AIPD2", funded by the European Regional Development Fund in the Republic of Croatia under the Operational Programme Competitiveness and Cohesion 2014 – 2020.

References

[1] J. Camacho-Collados and M. T. Pilehvar, 'From Word To Sense Embeddings: A Survey on Vector Representations of Meaning', *Journal of Artificial Intelligence Research*, vol. 63, pp. 743–788, Dec. 2018, doi: 10.1613/jair.1.11259.

[2] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, 'Text Classification Algorithms: A Survey', *Information*, vol. 10, no. 4, p. 150, Apr. 2019, doi: 10.3390/info10040150.

[3] S. Ruder, I. Vulić, and A. Søgaard, 'A Survey Of Cross-lingual Word Embedding Models', *jair*, vol. 65, pp. 569–631, Aug. 2019, doi: 10.1613/jair.1.11640.

[4] S. Wang, W. Zhou, and C. Jiang, 'A survey of word embeddings based on deep learning', *Computing*, vol. 102, no. 3, pp. 717–740, Mar. 2020, doi: 10.1007/s00607-019-00768-7.

[5] Y. Wang, Y. Hou, W. Che, and T. Liu, 'From static to dynamic word representations: a survey', *Int. J. Mach. Learn. & Cyber.*, vol. 11, no. 7, pp. 1611–1630, Jul. 2020, doi: 10.1007/s13042-020-01069-8.

[6] Y. Bengio, R. Ducharme, and P. Vincent, 'A Neural Probabilistic Language Model', in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 932–938.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, 'Natural Language Processing (almost) from Scratch', *arXiv:1103.0398 [cs]*, Mar. 2011, Accessed: Oct. 29, 2020. [Online]. Available: http://arxiv.org/abs/1103.0398.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, 'Efficient Estimation of Word Representations in Vector Space', *arXiv:1301.3781 [cs]*, Sep. 2013, Accessed: Oct. 27, 2020. [Online]. Available: http://arxiv.org/abs/1301.3781.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, 'Distributed Representations of Words and Phrases and their Compositionality', in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[10] J. Pennington, R. Socher, and C. Manning, 'GloVe: Global Vectors for Word Representation', in *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1532–1543, doi: 10.3115/v1/D14-1162.

[11] T. Luong, R. Socher, and C. Manning, 'Better Word Representations with Recursive Neural Networks for Morphology', in *Proc. 17th Conference on Computational Natural Language Learning*, Sofia, Bulgaria, Aug. 2013, pp. 104–113, Accessed: Oct. 29, 2020. [Online]. Available: https://www.aclweb.org/anthology/W13-3512.

[12] P. Gage, 'A New Algorithm for Data Compression', *The C Users Journal*, vol. 12, no. 2, pp. 1–14, 1994.

[13] R. Sennrich, B. Haddow, and A. Birch, 'Neural Machine Translation of Rare Words with Subword Units', *arXiv:1508.07909 [cs]*, Jun. 2016, Accessed: Oct. 23, 2020. [Online]. Available: http://arxiv.org/abs/1508.07909.

[14] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, 'Bag of Tricks for Efficient Text Classification', *arXiv:1607.01759 [cs]*, Aug. 2016, Accessed: Oct. 29, 2020. [Online]. Available: http://arxiv.org/abs/1607.01759.

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, 'Enriching Word Vectors with Subword Information', *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

[16] B. Heinzerling and M. Strube, 'BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages', presented at the LREC 2018, Miyazaki, Japan, May 2018, Accessed: Oct. 23, 2020. [Online]. Available: https://www.aclweb.org/anthology/L18-1473.

[17] C. N. Dos Santos and B. Zadrozny, 'Learning character-level representations for part-of-speech tagging', in *Proc. 31st ICML*, vol. 32, Beijing, China, Jun. 2014, p. II-1818-II–1826, Accessed: Nov. 10, 2020. [Online].

[18] X. Chen, L. Xu, Z. Liu, M. Sun, and H. Luan, 'Joint Learning of Character and Word Embeddings', in *Proceedings of IJCAI'15*, pp. 1236–1242, 2015.

[19] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, 'Character-Aware Neural Language Models', *arXiv:1508.06615 [cs, stat]*, Dec. 2015, Accessed: Oct. 23, 2020. [Online]. Available: http://arxiv.org/abs/1508.06615.

[20] T. Rama and Ç. Çöltekin, 'LSTM Autoencoders for Dialect Analysis', in *Proc. 3rd Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, Osaka, Japan, Dec. 2016, pp. 25–32, Accessed: Oct. 27, 2020. [Online]. Available: https://www.aclweb.org/anthology/W16-4803.

[21] M. E. Peters *et al.*, 'Deep contextualized word representations', *arXiv:1802.05365 [cs]*, Mar. 2018, Accessed: Oct. 23, 2020. [Online]. Available: http://arxiv.org/abs/1802.05365.

[22] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, 'Improving Language Understanding by Generative Pre-Training', preprint, p. 12, May 2018.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', *arXiv:1810.04805 [cs]*, May 2019, Accessed: Oct. 27, 2020. [Online]. Available: http://arxiv.org/abs/1810.04805.

[24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, 'XLNet: Generalized Autoregressive Pretraining for Language Understanding', *arXiv:1906.08237 [cs]*, Jan. 2020, Accessed: Jan. 31, 2021. [Online]. Available: http://arxiv.org/abs/1906.08237