International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4s
DOI: https://doi.org/10.17762/ijritcc.v11i4s.6440
***Article Received**: 12 January 2023 **Revised**: 20 February 2023 **Accepted**: 27 February 2023*

An Overview of Context Capturing Techniques in NLP

Dhawal Khem[*1], Shailesh Panchal[2], Chetan Bhatt[3]
[*1]Ph.D. Scholar, Computer Engineering,
Gujarat Technology University, GTU,
Ahmedabad(Gujarat), India
e-mail: khemdhawal@gmail.com
ORCID ID : 0000-0002-8064-5954
[2]Professor, PG-Cyber security,
Graduate School of Engineering & Technology (GSET)
Ahmedabad(Gujarat), India
e-mail: sdpanchal@gtu.edu.in
[3]Professor, Instrumentation and Control Engineering,
MCA College Maninagar, K. K. Shastri Educational Campus,
Khokhra Road, Ahmedabad(Gujarat), India
e-mail: chetan_bhatt@yahoo.com

Abstract—In the NLP context identification has become a prominent way to overcome syntactic and semantic ambiguities. Ambiguities are unsolved problems but can be reduced to a certain level. This ambiguity reduction helps to improve the quality of several NLP processes, such as text translation, text simplification, text retrieval, word sense disambiguation, etc. Context identification, also known as contextualization, takes place in the preprocessing phase of NLP processes. The essence of this identification is to uniquely represent a word or a phrase to improve the decision-making during the transfer phase of the NLP processes. The improved decision-making helps to improve the quality of the output. This paper tries to provide an overview of different context-capturing mechanisms used in NLP.

Keywords- Natural Language Processing, Contextualisation, Word Representation, Word Embedding.

I. INTRODUCTION

Natural languages are context-sensitive languages, meaning the interpretation of a word or a phrase depends upon the surrounding words within which the word or phrase has been used. The surrounding word represents the context or the situation in which the specific word or phrase has a certain specific meaning. The context can be defined as the background or a frame that specifies the appropriate interpretation. If the context is not identified properly it may lead to wrong interpretation of the word or the phrase. Further, this may lead to miscommunication. If the context is captured properly it helps to deal with the ambiguous situation, where there is more than one different interpretation for the same word or phrase. The context capturing reduces the situation of ambiguity during decision-making in various NLP tasks, such as text translation[1], text simplification[2], information retrieval[3], etc.

Context capturing is also known as context analysis. It has two definitions in NLP. The first definition says context capturing is to split the sentences into groups of n-grams, noun phrases, themes, and facets. The second definition says context capturing is a process of deriving background information. According to this, the NLP processes like POS tagging[3], Named Entity Recognition[4], Dependency Parsing[5], etc. all come under context capturing techniques, which try to capture the background information related to the sentence.

Al-Thanyyan et. al. 2021 [6] presented an extensive study in the field of text simplification and gave a survey on various text simplification mechanisms along with its evaluation methods. Khem et. al. 2023 [7] showed an experimental setup for improving text translation from Gujarati regional language to English language by using text simplification in the preprocessing phase. To improve text translation further we can use context embedding layers to the text simplification. Here in this paper various context analysis techniques are discussed. .

II. CONTEXT ANALYSIS TECHNIQUES

Based on splitting the sentences into groups of words to extract the context, there are four methods of context analysis.

A. n-gram Extraction

The N-grams or n-words are combinations of one or more words that represent entities, phrases, concepts, and themes that appear in the text. The ‘n’ in ‘n-grams’ is the number of words

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4s
DOI: https://doi.org/10.17762/ijritcc.v11i4s.6440
Article Received: 12 January 2023 Revised: 20 February 2023 Accepted: 27 February 2023

in the word-group. The n-grams can be subdivided into mono-gram, bi-gram, and tri-grams based on the value of 'n'. The smaller the value of 'n' the group of n-grams represents a generalized phrase or entity. The higher the value of 'n' the n-grams represent a more specific phrase or entity. The 1-grams known as mono-grams are used to extract the entities and themes. The 3-grams or the tri-grams are used for phrase extraction. The mono-gram is too generalized and the tri-gram is too specific for context identification. The 2-grams or the bi-grams are used for context analysis. The drawback or limitation of n-gram is, the n-gram requires a long list of stop words to extract meaningful words, and always the n-gram group of words does not need to indicate the important text.

B. Phrase Extraction

A group of words that indicates parts of speech (POS) patterns are known as phrases. The phrases are made up of nouns, verbs, and other POS patterns. The noun phrases are extracted to identify what is being discussed. And the verb phrases are extracted to identify what is being done. To extract these phrases stop words are required, and compared to n-gram it requires less effort. The drawback of phrase extraction is, phrase extraction is limited to the words within the text, and there is no way to resolve the semantic ambiguity if it occurs, so deciding which phrase is contextually more relevant than the other is an unsolved problem.

C. Themes Extraction

Themes are noun phrases with contextual relevance scores. These themes are identified and extracted based on part of speech patterns and then they are scored with a lexical chaining process. Lexical chaining is a text analysis process that joins sentences through related nouns. The drawback of theme extraction and scoring is, it is limited to words within the text. Latent semantic analysis (LSA) techniques in NLP mathematically determine themes within text.

D. Facets Extraction

Facets can be defined as a specific perspective or viewpoint of an entity with many angles. It is different from the aspect which defines the situation. Facets are called smart filters, they are used to narrow down the search results. Facets can be static or dynamic. They can be set up for every search query, or they can change depending on the context of the query. Facets are used for review and survey processing.

III. CONTEXTUALIZED WORD REPRESENTATION

In NLP there are two ways of word representation:

1) Word Encoding, and 2) Word Embedding.

A. Word Encoding

Text processing is a non-trivial task for a machine as machines do not understand text directly. Machines understand numbers and can perform various mathematical operations on numbers. Thus representing a text with a unique number is required. Assigning a unique number to a text is called encoding. In the encoding process, a word is converted into a number or a vector of numbers. The vector of numbers represents or preserves the context and relationship between words and sentences. The encoding enables the machine to understand the pattern associated with the text and can identify the context of sentences.

- ***State Encoding:***

  A finite state machine (FSM) is a state model of a system. Where each state has a label or a name. In the state technique, the entire vocabulary of a language is considered as a set of strings where each string represents a state of FSM. Here based on the FSM there are two state encoding techniques used:

  One-Hot Encoding: In this encoding, each FSM state is defined as a separate bit. So if an FSM has N states N bits are required to represent each state. The name one-hot is used because for a word only one bit is "hot" or TRUE at any time. One hot encoding requires more flip-flops compared to binary encoding as each bit of a state gets stored in a flip-flop. The text-to-number and the number-to-text conversion processes are simple with One-hot encoding thus it requires less number of gates.

  Binary Encoding: In this encoding, each state of FSM is defined using a binary number, as a combination of fixed-length bits. So FSM containing N states requires a binary number with $\log_2 N$ bits to uniquely represent each state. Compared to the One-Hot encoding in binary encoding, it requires less number of flip-flops and more gates. Thus the choice of using either One-Hot encoding or binary encoding depends on the FSM used.

- ***TF Encoding:*** The TF is a short form of the word Term Frequency. Here the term is used for a word or a phrase of a text. The TF is defined as the probability or the percentage of term occurrence, i.e. the ratio of the total number of times a term occurs in a document, to the total number of terms used in the document. The mathematical equations for calculating TF are:

  $$TF(t) = F_D(t)/N_D$$

  where t is the term or the word in the text. F_D is the frequency of the corresponding term t in document D,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4s
DOI: https://doi.org/10.17762/ijritcc.v11i4s.6440
Article Received: 12 January 2023 Revised: 20 February 2023 Accepted: 27 February 2023

and N_D is the total number of terms used in document D, i.e. the vocabulary list in use.

- ***TF-IDF Encoding:*** The TF is the same discussed earlier, the Term Frequency. and the IDF is a short form of the Inverse Document Frequency. It can break a word into two parts: TF and IDF. The mathematical equations for calculating TF-IDF are:

$$TF\text{-}IDF(t) = TF(t) * IDF(t)$$

where the t is the term or the word in the text. The TF(t) is defined earlier, and the IDF is defined as:

$$IDF(t) = \log(D(t)/T_D+1)$$

where D(t) is the number of documents containing the term t. The D is the document or the vocabulary list in use. T_D the number of documents containing the term t.

The IDF calculated how often the term appears in all documents. The IDF decreases the weight of the frequently used terms and increases the weight of the rarely used terms. Table I shows an example of TF-IDF encoding.

TABLE I. TF-IDF ENCODING

word	count	tf	idf	tf-idf
modicum	1	0	1.012	0
model	3	1	0.095	0.095
modeling	1	1	0.201	0.201
models	2	1	0.201	0.201

B. Word Embeddings

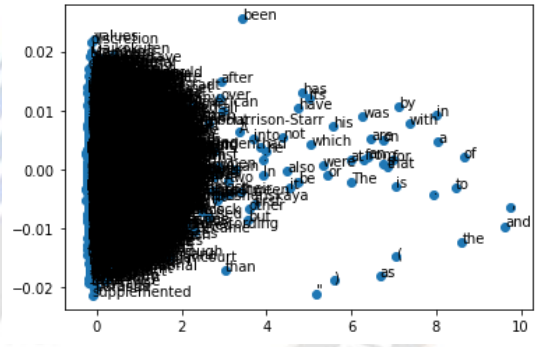
Word embedding is a vectorial representation of a word. The collection of N-dimensional vectors for each word in a vocabulary builds an N-dimensional vector space. The N values in a vector are real values which are the corresponding distance in that specific dimension in the vector space. From that perspective, it is said that the vector preserves the meaning of the word, i.e. the vectors which are closer in the vector space, are similar in meaning. The higher the dimension the more complex the vector space becomes and requires higher computational power. The dimensions are usually latent, and often obtained using the information as in the distributional semantics approach (e.g. LSA, word2vec).

Word Vector: It vectorised representation of a word or a phrase. The vectors consist of a series of real valued numbers, in which each real value represents the distance of a data point in that specific dimension in vector space. Below is an example of a word vector for the word 'armed'.

```
word='armed'
vector=[
-0.01516168, 0.04691967, 0.04435506, 0.04524148, 0.00314969, -0.10343689, 0.03699014, 0.12269781, -0.04992141, -0.07716558, -0.00292364, -0.06410118, -0.02454621, 0.04738826, -0.0021141, -0.02485863, 0.06427551, -0.02007342, 0.05281805 ,-0.05272718, -0.02216516, 0.01460739, …]
```

These vectors are usually used to capture some notion of similarity through dot products or other vectorial operations. Collectively many vectors build a vector space. A vector within a vector space is a single dot representing a data point.

Vector Space: The vector space is a multidimensional space within which words and phrases are denoted as a vectorised data point. The vectors are further analyzed through various mathematical models to derive the relationship among the related data points. The related data points are closer in proximity compared to the unrelated data points. Figure 1 shows a sample of a vector space.



Word2Vec<vocab=12014, vector_size=100, alpha=0.025>

Figure 1. Vector Space

In various applications such as information retrieval [3], document classification [8], question answering [9], named entity recognition [10], and parsing [11], the vectors are used as features.

IV. STATIC WORD EMBEDDINGS MODELS

It is a neural network(NN) model used to produce word embedding. The number of layers used in such NN models is very few. It converts the words into their corresponding vector form. It takes a large dataset containing the language vocabulary as input and produces the corresponding output vector space. Each word in the vocabulary maps to a unique vector.

- **Word2Vec[5]** This is a static embedding model. It maps a word with its context. The Continuous Bag of Words (CBOW) is a type of Word2Vec model that predicts the word from the given context. The Skip-

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4s
DOI: https://doi.org/10.17762/ijritcc.v11i4s.6440
Article Received: 12 January 2023 Revised: 20 February 2023 Accepted: 27 February 2023

Gram is a type of Word2Vec model that predicts the context based on the given word. The Word2Vec model is suitable for smaller databases. So it requires less storage space.

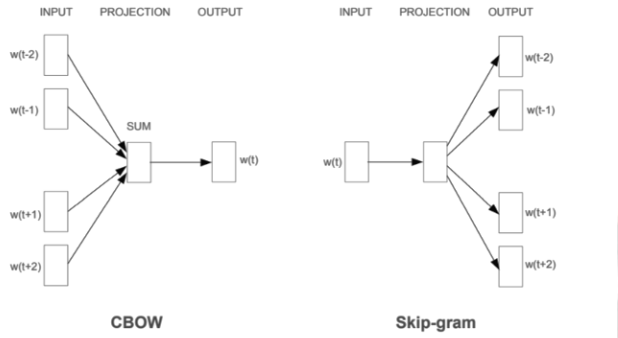


Figure 2. CBOW and Skip-gram[12]

- **GloVE[5]:** Global Vectors (GloVE) is a model obtained from an unsupervised algorithm for distributed vectorial representation of words. The semantic similarity between words can be obtained by the distance between words in the vector space. Pennington et al., 2014 [5] and Bojanowski et al., 2017 [13] showed improvements in the results shown by Mikolov et al. [14].
- **Sent2Vec:** It is a wrapper on Word2Vec to obtain a vector of a sentence. To obtain the vector of a sentence, an average sum of each vector of a word in the sentence is taken. Pagliardini et al., 2017 [15] used this idea to generate the sentence vectors.
- **Universal Sentence Embeddings:** This is an encoder model that encodes textual data into high-dimensional vectors. It specifically targets transfer learning to other NLP tasks, such as text classification, semantic similarity, and clustering. Cer et al., 2018 [16] created universal sentence vectors using transformers and DANs.
- **Doc2Vec**: It is another widely used technique. It computes a feature vector for every document in the corpus. Le and Mikolov, 2014 [17] used this technique to compute the feature vector for every document in the corpus.

Static word embedding models are context-independent models. They produce one vector embedding for each word, by combining different contexts of the word usage in one vector representation.

V. DYNAMIC WORD EMBEDDINGS MODELS

The limitation of the earlier models is the vector representation for each word is fixed, and does not change with its corresponding usage or context. To overcome this limitation, contextual word embedding models were created. These models learn the "senses" dynamically. The vector representation of a word changes dynamically based on the context in which the word is used. Recently various language models have used neural networks based on transformer architecture to build contextualized dynamic word embeddings [18, 19, 20, 21]. The dynamic word embedding models are further categorized into two groups based on the training corpus:

A. Monolingual Models

In such words embedding models, the models are trained in a single language. Following are the recent monolingual word embedding models,

- **Context2vec:** This is an unsupervised multi-layer perceptron model. It uses a bi-directional LSTM (Long Short Term Memory) context-preserving model to extract the word embedding of a word as per the context.
- **ELMO:** Embeddings from Language Model (ELMO) is a character based convolution model for sequence learning. It can handle out of vocabulary words. Peters et al., 2018 [19] used bidirectional LSTMs to build an improved ELMO model.
- **BERT:** Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based semi-supervised sequence learning architecture created by Andrew Dai and Quoc Le. Devlin et al., 2018 used BERT to learn the bidirectional context and be able to establish a state-of-the-art across different tasks. XLNet[22] improved earlier BERT by using a permutation language model to address BERT issues and surpassed the BERT on several tasks.

B. Cross-lingual word embeddings

In cross-lingual word embedding models two monolingual vector spaces are used to learn the common projection. It has shown usefulness in several cross-lingual tasks such as information extraction [23], False Friends and Cognate detection [24], and Neural networked-based unsupervised machine translation [25].

- **MUSE**: Multilingual Unsupervised and Supervised Embeddings (MUSE) is a Python library that enables faster and easier development and evaluation of cross-lingual word embeddings and natural language processing. Conneau et al., 2017 [26] showed the usage of cross-lingual embeddings across different languages.
- **VecMap[26]:** It is an open-source cross-lingual word embedding implementation. Artetxe et al., 2018 [27] trained a BERT in a multilingual fashion and

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4s
DOI: https://doi.org/10.17762/ijritcc.v11i4s.6440
***Article Received**: 12 January 2023 **Revised**: 20 February 2023 **Accepted**: 27 February 2023*

introduced an unsupervised learning method for these embeddings.

- **XLM:** Cross-lingual Language Model (XLM) uses a dual-language BERT model with a Byte Pair Encoding(BPE) pre-processing technique to learn relations between words in different languages. Lample and Conneau, 2019 [28] showed an improved BERT in the cross-lingual platform.
- **FastText**: It is an extension of the word2vec model. It represents each word as an n-gram of characters. It has several pre-trained words embedded in multiple languages.

ELMo has shown improvement in NLP for low-resource languages [29]. Similar methods which are less expensive in computations have the potential for bigger use [30].

A recent study showed no matter how well sense and contextualized representations capture word meaning in context [31], the state-of-the-art sense and contextualized representation techniques failed in accurately distinguishing meanings in context. It performed slightly better than the baseline. The meaning conflation or the semantic ambiguities remains an open and unsolved problem.

VI. Conclusion

In this paper, we discussed context capturing techniques. Some techniques are based on the text analysis through text pre-processing and some are through building a trained model. Some techniques are based on the word representation using a single value and some are based on a series of values known as a vector. Some techniques are based on the static embedding approach and some are based on the dynamic embedding approach. Some techniques are based on usage of the monolingual dataset and some are based on usage of multilingual dataset. The context capturing technique can be broadly categorized into two categories: context-independent and context-dependent .

Word2vec and Glove word embeddings are context-independent. These models output just one vector (embedding) for each word, regardless of where the words occur in a sentence and regardless of the different meanings they may have. The ELMo, and BERT word embedding are context dependent models. They use a dynamic model to build the word embedding for the word based on their context. The key difference between the BERT and the other models is the way of generating the embeddings. In Glove and Word2vec are word-based models, the models take input words and output word embeddings. While the Elmo model is a character-based convolution model, thus it can handle out of vocabulary words. The BERT is a subwords based dynamic word embedding model. It represents input as subwords and learns embeddings for the subwords. So it requires a very small vocabulary compared to the Glove, Word2vec, or ELMo model. Word embedding models are low-dimensional vector space compared to the high-dimensional distributional semantics approach. The cross lingual context capturing models are discussed, which shows improvement in the state of art performance of monolingual BERT models.

At the end a recent study showed no matter how dynamic and flexible word embedding models have become in capturing the contextual properties of the words, still the word embeddings are yet hampered by the meaning conflation deficiency, which is an ambiguity at the semantic level to discriminate different meanings of a word.

References

[1] Modh, Jatin C. “A STUDY OF MACHINE TRANSLATION APPROACHES FOR GUJARATI LANGUAGE.” International Journal of Advanced Research in Computer Science 9, no. 1 (February 20, 2018): 285–88. https://doi.org/10.26483/ijarcs.v9i1.5266.

[2] Siddharthan, Advaith (28 March 2006). "Syntactic Simplification and Text Cohesion". Research on Language and Computation. 4 (1): 77–109. doi:10.1007/s11168-006-9011-1. S2CID 14619244

[3] Manning, Christopher D. “Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?” In Computational Linguistics and Intelligent Text Processing, edited by Alexander F. Gelbukh, 6608:171–89. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. https://doi.org/10.1007/978-3-642-19400-9_14.

[4] Sebastiani, Fabrizio. “Machine Learning in Automated Text Categorization.” ACM Computing Surveys 34, no. 1 (March 2002): 1–47. https://doi.org/10.1145/505282.505283.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of EMNLP, pages 1532–1543. https://nlp.stanford.edu/projects/glove/

[6] Al-Thanyyan, Suha S., and Aqil M. Azmi. “Automated Text Simplification.” ACM Computing Surveys 1 Apr. 2021. ACM Computing Surveys. Web..

[7] Dhawal Khem, Shailesh Panchal, Chetan Bhatt, “Text Simplification Improves Text Translation from Gujarati Regional Language to English: An Experimental Study”, Int J Intell Syst Appl Eng, vol. 11, no. 2s, pp. 316–327, Jan. 2023. VBNCVX

[8] Sebastiani, Fabrizio. “Machine Learning in Automated Text Categorization.” ACM Computing Surveys 34, no. 1 (March 2002): 1–47. https://doi.org/10.1145/505282.505283.

[9] Tellex, Stefanie, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. “Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering,” n.d.

[10] Turian, Joseph, Lev Ratinov, Y. Bengio, and Dan Roth. “A Preliminary Evaluation of Word Representations for Named-Entity,” January 1, 2009.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 4s
DOI: https://doi.org/10.17762/ijritcc.v11i4s.6440
***Article Received**: 12 January 2023 **Revised**: 20 February 2023 **Accepted**: 27 February 2023*

[11] Socher, Richard, John Bauer, Christopher D. Manning, and Andrew Y. Ng. "Parsing with Compositional Vector Grammars." In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 455–65. Sofia, Bulgaria: Association for Computational Linguistics, 2013. https://aclanthology.org/P13-1045.

[12] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association of Computational Linguistics, 5(1):135–146. https://github.com/facebookresearch/fastText

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. CoRR abs/1301.3781. https://code.google.com/archive/p/word2vec/

[14] Pagliardini, Matteo, Prakhar Gupta, and Martin Jaggi. "Unsupervised Learning of Sentence Embeddings Using Compositional N-Gram Features." In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 528–40. New Orleans, Louisiana: Association for Computational Linguistics, 2018. https://doi.org/10.18653/v1/N18-1049.

[15] Cer, Daniel, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, et al. "Universal Sentence Encoder." arXiv, April 12, 2018. https://doi.org/10.48550/arXiv.1803.11175.

[16] Le, Quoc V., and Tomas Mikolov. "Distributed Representations of Sentences and Documents." arXiv, May 22, 2014. https://doi.org/10.48550/arXiv.1405.4053.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. https://arxiv.org/abs/1810.04805

[18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In Proceedings of NAACL, New Orleans, LA, USA, pages 2227–2237. https://allennlp.org/elmo

[19] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning generic context embedding with bidirectional LSTM. In Proceedings of CoNLL, Berlin, Germany, pages 51–61.

[20] Lample, Guillaume, and Alexis Conneau. "Cross-Lingual Language Model Pretraining." arXiv, January 22, 2019. https://doi.org/10.48550/arXiv.1901.07291.

[21] Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." arXiv, January 2, 2020. https://doi.org/10.48550/arXiv.1906.08237.

[22] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 333–342. Association for Computational Linguistics.

[23] Merlo, Paola, and Maria Andueza Rodriguez. "Cross-Lingual Word Embeddings and the Structure of the Human Bilingual Lexicon." In Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), 110–20. Hong Kong, China: Association for Computational Linguistics, 2019. https://doi.org/10.18653/v1/K19-1011.

[24] Artetxe, Mikel, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. "Unsupervised Neural Machine Translation." arXiv, February 26, 2018. https://doi.org/10.48550/arXiv.1710.11041.

[25] Conneau, Alexis, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. "XNLI: Evaluating Cross-Lingual Sentence Representations." arXiv, September 13, 2018. https://doi.org/10.48550/arXiv.1809.05053.

[26] Artetxe, Mikel. "VecMap (Cross-Lingual Word Embedding Mappings)." Python, March 15, 2023. https://github.com/artetxem/vecmap.

[27] Lample, Guillaume, and Alexis Conneau. "Cross-Lingual Language Model Pretraining." arXiv, January 22, 2019. https://doi.org/10.48550/arXiv.1901.07291.

[28] Mulcaire, Phoebe, Jungo Kasai, and Noah A. Smith. "Low-Resource Parsing with Crosslingual Contextualized Representations." arXiv, September 18, 2019. https://doi.org/10.48550/arXiv.1909.08744.

[29] Gururangan, Suchin, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks." arXiv, May 5, 2020. https://doi.org/10.48550/arXiv.2004.10964.

[30] Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. Wic: 10,000 example pairs for evaluating context-sensitive representations. arXiv preprint arXiv:1808.09121. https://arxiv.org/abs/1808.09121

[31] Kulshrestha, Ria. "NLP 101: Word2Vec — Skip-Gram and CBOW." Medium, October 26, 2020. https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314.