"Gimme a break": A Study on Contextual Word Embedding based Homonym Detection

Devashish Kamble & Ecesu Ürker

February 2023

Abstract

Homonym identification is important for Word Sense Disambiguation (WSD) that requires coarse grained partitions of senses. This project aims to indicate whether contextual information is sufficient for identifying a homonymous word. To capture the context, BERT embeddings are used as opposed to Word2Vec or GloVe, which consolidate senses into one vector. The WiC dataset is leveraged for the wide range of homonymous sentences it holds. In order to check for homonymity, we have used cosine similarity as a metric for measuring the closeness of the word senses in three different sentences and make a judgement based on the comparison.

1 Introduction and Motivation

In the past decade, word embeddings became popular in research on lexical semantics. Word2vec [1] being introduced as one of the pioneering word embedding models, caused drastic changes in the field of lexical semantics. However, these approaches have a crucial limitation, that their word representations are context-insensitive and they do not take into account the meaning derived from the surrounding words. Thus, the so-called static models rely only on the word and generate a single embedding for words representing different senses or meanings. In this study, we focus on one such category of ambiguous words called **Homonyms**. Homonymy refers to the cases where different senses of a word are completely unrelated, e.g the word "bank" has the senses of *sloping land* and *financial institution* with the homonymy relation. Thus, words with semantically unrelated senses, such as "bank", are called homonyms. Furthermore, the term *lexeme* is used to refer to a pair of words and its meaning in a lexicon. Each lexeme covers a set of semantically related senses of a word. Therefore, a homonymous word has more than one lexeme. Using static models results in the assignment of the exact same vector representation for a pair of homonyms, like the words "calf" (back of the leg below the knee) and "calf" (young cow or bull), representing different senses. Among the most widely used models like GloVe [2], the problem associated with homonyms has been identified as a weakness.

Then comes into picture the context-based models, which generate embeddings based on the semanticity of not just the word but its surroundings as well. Thus, using this approach, in which all words in the text have a different embedding vector based on their context is crucial for solving the issue of word sense disambiguation. The contextual embeddings in these methods should allow different meanings of homonyms to be represented in different vectors, in turn overcoming the shortcomings of the static models. In this study, the performance of context-based methods is put into test for representing homonyms differently and is evaluated using a recent state-of-the-art model, **BERT**.

2 Literature and related work

The words in our language are human-readable but not by machines, and hence need to be converted into vectors of numbers through an embedding procedure. It is important for the vectors to encapsulate the syntax and semantics of the words holistically, in order to make precise inferences. GloVe [2] and Word2Vec [1] are the most frequently used methods of word embedding but these are static or context-insensitive. As an alternative, contextual word-embedding models like ELMo [3], BERT [4], and GPT-3 [5] provide a viable option, as they take into account the context of the entire input sentence.

There have been a number of studies on homonym word detection. Liu et al. [6] used a three-step clustering for name disambiguation utilising common coauthors. Shen et al. [7] developed a system that utilises user feedback for disambiguation. Finally, Pittke et al. [8] proposed a technique that detects lexical ambiguities such as homonyms in a process model. Mridha et al. [9] used a self-developed confused work list by editing the distance and applying a naive Bayes classifier to detect typographical errors and homonyms. Also, An et al. [10] proposed several algorithms for detecting homonym errors, using the hash method to efficiently calculate the distance.

Most of the previous work for homonym detection uses either a rule-based or statistical-based approach which when applied to a general text domain would fail as they are limited to a specific text domain. But there are a few studies that take into account the semantics, one of them being the work of van den Beukel et al. [11] where humour in short text is recognized by integrating homophone and homograph based features into existing humour recognition systems. Another study talks about the importance of differentiating polysemy and homonymy in information retrieval using morphology, part-of-speech, and phrases [12]. The work of Wiedemann et al. [13] uses contextual embeddings from BERT, ELMo and Flair NLP to classify WordNet senses using a supervised classification algorithm (K-Nearest Neighbours).

However, unfortunately in the previous literature the studies conducted with contextualised models have given out mixed results. Saha R. [14] conducted an interesting study in which clustering algorithms were applied on the BERT embedding but in his study, the contextual embeddings failed to detect the homonyms. Contrarily, Garcia M. [15] conducted a study in which he compared different contextualised and static models, and found that contextualised models were able to detect the homonyms while the static models failed to do so. With our study we hope to provide more evidence toward either of the sides using BERT as a common ground.

3 Research question and Proposed Solution

As aforementioned in the prior research, there are a few conflicting views on the usage and results produced by using the contextualised word embeddings when applying the BERT model. Our study aims to disentangle this issue. Hence, the principal question in our concern is accordingly, *"Are contextualised word embeddings generated through BERT sufficient for precise detection of homonyms?"*.

To answer this question, we have given BERT three sentences in which two of them had the same sense of a homonymous word and one of them had another sense and asked BERT to compare the word vectors for the homonym word in each sentence. Then depending on the BERT's evaluations, we calculated the error rate.

4 Our Project

4.1 Model

BERT, or Bidirectional Encoder Representations from Transformers, improves upon standard Transformers by removing the unidirectionality constraint by using a masked language model (MLM) pre-training objective. The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike left-to-right language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows to pre-train a deep bidirectional Transformer. In addition to the masked language model, BERT uses a next sentence prediction task that jointly pre-trains text-pair representations [16].

We are using the pre-trained BERT model provided by **HuggingFace Transformers** [4], as they are easier to use and the architecture is provided for free. Additionally, they typically have better results and require less or no training (zero-shot classification). Google released a few variations of BERT models and we are using the 'base' model out of two available sizes ('base' and 'large') which ignores casing, hence is 'uncased'. The model consists of twelve layers [16].

4.2 Method for Homonym Detection

In our study, we are using **Google Colab** [17] as it allows us to execute Python code through the browser and is an excellent tool for deep learning tasks. It is a hosted Jupyter notebook that requires no setup and gives free access to Google computing resources such as GPUs and TPUs. The BERT model can take as input either one or two sentences, and uses the special token [SEP] to differentiate them. The [CLS] token always appears at the start of the text, and is specific to classification tasks. We are passing one sentence at a time to generate embeddings for each. Next, the model tokenizes all the words, wherein, it first checks if the whole word is in its vocabulary. If not, it tries to break the word into the largest possible subwords known to it or will either disintegrate the word into individual characters. When words are split into subwords the embeddings for each of them are averaged and assigned to the original word [16]. After breaking the text into tokens, the sentence goes from a list of strings to a list of vocabulary indices. BERT needs the segment token for a sentence; as we have a single-sentence input, it requires a series of 1s, so we created a vector of 1s for each token in our input sentence. We make use of the PyTorch Framework [18] to generate tensors based on the preprocessed data. On passing these tensors as input to BERT, the output is a four dimensional object:

1. Number of **layers**(13)
2. Number of **sentences**(1)
3. Number of **tokens** in the sentence
4. Number of **features** (768)

Thus, for each token of our input we have **13** separate vectors each of length **768**. The BERT authors tested word-embedding strategies by feeding different vector combinations as input features on a Named Entity Recognition (NER) task and observed that concatenation of the last four layers produced the best results on this specific task [19], followed by results produced by sum of the last four layers. We have, therefore, decided to go with the **concatenation approach** in our study.

To test our model, we used **WiC dataset** which is initially designed as a binary task for word embedding models. It consists of sentence pairs that either have the same or different senses of a homonymous word [20]. For this study, we specifically used sentence pairs that are related to the different senses of the same word to test the ability of our model to detect homonymity. Since our task required a third sentence for comparison, we have generated an additional sentence for the 50 sentence pairs. These sentences were generated in a way that had the sense of the word that was present in the first sentence. The dataset and code is available to on GitHub [1].

Homonym	**I1**	**I2**	**I3**	**S1**	**S2**	**S3**
carry	0	2	1	Carry your camping gear.	The student carries his bag.	Sound carries well over water.
go	2	3	2	Messages must go through.	I really must go	Will sofa go through it?
break	0	1	2	Break an alibi.	He broke the pattern	Can you break it down?

Table 1: Sample data used in the study

Then, we calculated contextualised word embedding for each sentence and based on the outcome, we made a semantic comparison between the sentences. Comparisons are made by calculating the **cosine similarity**. Mathematically, cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors would be the tensors obtained for homonymous words present in the sentences. Thus, we find the similarity between the two sentences having words with close senses, which should yield a higher cosine similarity value when compared to either of the other sentences having words with different senses and hence conclude that word is a true homonym. Below is an example of the way in which we are detecting homonyms:

cosine	**S1**	**S2**	**S3**
S1	1		
S2	0.48	1	
S3	0.39	0.25	1

Table 2: S1: Messages must go through. S2: I really must go S3: Will sofa go through it?

As the cos(S1,S2) > cos(S1,S3) and also cos(S1,S2) > cos(S2,S3), we deduce that *go* in this case is a true homonym.

5 Results

On performing the similarity judgement on the first 50 sentence triple pairs, we found that word embeddings generated by BERT were able to distinguish homonyms present in 26 triple pairs which corresponds to a 48% error rate. The below table summarises the observed results:

True Positive	26	Error Rate = 48%
True Negative	24	
Total Sentence Triple Pairs	50	

Table 3: Results

[1] github.com/ecesuurker/Homonym-Detection-System

6 Discussion

In this study, we investigated the task of homonym identification wherein we made an attempt at resolving the banter around BERT's performance to check if the contextual information in the form of word embeddings is sufficient for identifying homonyms. We found that for a total of 50 sentence triple pairs, BERT came through with an error rate of 48%. Even though we might conclude that the baseline model in our case accomplished the task, it did so by a small margin.

There could be a few ways by which we could tweak the performance: In our study, we make use of word-level embeddings but there is a possibility of improving the results through the usage of sentence-level embeddings as the authors of WiC dataset found better results for the homonym detection when the sentence-level embeddings were used. Moreover, in our study we used the base BERT model, instead replacing it with the "large" BERT version would enhance the performance in homonym detection tasks [20]. Also instead of just utilising the method of concatenating the last four layers, we could generate embeddings from the sum of the last four layers or second-to-last layer only.

Furthermore, the dataset can also be improved by considering additional homonymous sentence pairs, as the results generated from 50 sentences might not be statistically significant. On the contrary, the authors of WiC dataset have claimed an accuracy of about 50% for any baseline model [20] and we were able to achieve it (52% in our case). In addition, the WiC database consists of sentence pairs that we had to generate a third sentence to test our model. There is a possibility that not all of the generated sentences were aligned with the sense of the first sentence which might have increased the error rate we have obtained. So, in the future studies, using a database that is fully compiled by linguistic experts or normed by the native speakers would give more reliable results.

Finally, we think the contextual models still have a long way to go in order to achieve human-level accuracy and development of such techniques will require intensive effort in terms of data collection and preparation, along with semantically intelligent learning algorithms.

7 Acknowledgement

We would like to thank **Prof. Rafaella Bernardi** for clarifying any questions and concerns during the course and sparking a keen interest in Computational Linguistics. We also appreciate her feedback about the project during project meetings, which helped to monitor our progress and address a few challenges. This project was also partly inspired by the coursework done in *Introduction to Human Language* by **Prof. Zamparelli**.

References

[1] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv*, 2013.

[2] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," *in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[3] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations. arxiv 2018," *arXiv preprint arXiv:1802.05365*, vol. 12, 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[6] W. Liu, R. Islamaj Doğan, S. Kim, D. C. Comeau, W. Kim, L. Yeganova, Z. Lu, and W. J. Wilbur, "Author name disambiguation for p ub m ed," *Journal of the Association for Information Science and Technology*, vol. 65, no. 4, pp. 765–781, 2014.

[7] Q. Shen, T. Wu, H. Yang, Y. Wu, H. Qu, and W. Cui, "Nameclarifier: A visual analytics system for author name disambiguation," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 141–150, 2016.

[8] F. Pittke, H. Leopold, and J. Mendling, "Automatic detection and resolution of lexical ambiguity in process models," *IEEE Transactions on Software Engineering*, vol. 41, no. 6, pp. 526–544, 2015.

[9] M. Mridha, M. A. Hamid, M. M. Rana, M. E. A. Khan, M. M. Ahmed, and M. T. Sultan, "Semantic error detection and correction in bangla sentence," in *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pp. 184–189, IEEE, 2019.

[10] Y. An, S. Liu, and H. Wang, "Error detection in a large-scale lexical taxonomy," *Information*, vol. 11, no. 2, p. 97, 2020.

[11] S. van den Beukel and L. Aroyo, "Homonym detection for humor recognition in short text," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 286–291, 2018.

[12] R. Krovetz, "Homonymy and polysemy in information retrieval," in *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 72–79, 1997.

[13] G. Wiedemann, S. Remus, A. Chawla, and C. Biemann, "Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings," *arXiv preprint arXiv:1909.10430*, 2019.

[14] R. Saha, "Cmput 600 project report: Homonym identification using bert,"

[15] M. Garcia, "Exploring the representation of word meanings in context: A case study on homonymy and synonymy," *arXiv preprint arXiv:2106.13553*, 2021.

[16] C. McCormick and N. Ryan, "Bert word embeddings tutorial," *URL: https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial*, 2019.

[17] E. Bisong, *Google Colaboratory*, pp. 59–64. Berkeley, CA: Apress, 2019.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[19] J. Alammar, "The illustrated bert, elmo, and co.(how nlp cracked transfer learning)[blog post]," 2018.

[20] M. T. Pilehvar and J. Camacho-Collados, "Wic: the word-in-context dataset for evaluating context-sensitive meaning representations," *arXiv preprint arXiv:1808.09121*, 2018.