Project Report

Andrea Bonaiuti mat. 1618825

Translate to Disambiguate: Zero-shot Multilingual Word Sense Disambiguation with Pretrained Language Models

1 Introduction

This report is based on a project regarding the task of Word Sense Disambiguation, more precisely the paper 'Translate to Disambiguate: Zero-shot Multilingual Word Sense Disambiguation with Pretrained Language Models' [3].
As the paper titles claims, this paper aims to leverage the task of Word Sense Disambiguation using Pretrained Language Models to translate words given in a context (a sentence).
The project consists in an overhaul rewriting of the project github, given its very confusing and bad-written state, resulting in an easier human readable framework with lot of comments explaining what it is done and a better use and distinction of methods and variable names.
Furthermore, it tries to improve the logic of the paper, correcting a behaviour during the translation phase and also proposing a different logic of the same translation phase.
The report will first present a little introduction of the task of interest, then a presentation of the paper logic. It continues going more in details of implementation, highlighting its weaknesses, some implementation decisions and their reasons. Then the paper logic under the inference details with its correction and finally a novel proposed method of inference. And as the last section a comparison and discussion over the results obtained.

2 Word Sense Disambiguation and Paper Method

WSD, that stands for Word Sense Disambiguation, is the task of identifying which sense of a word is meant in a sentence or other form of context.
This can be obtained through Knowledge-Base approaches using computational lexicons such as WordNet[2] and BabelNet[4], and graph algorithms like random walks[1] or game theory[6]; and through Supervised Learning methods, learning a parameterized function to map words in context to a sense from a predefined vocabulary.
Most of these methods make use of pretrained Transformers, especially the most recent works that reached the state-of-the-art performances, keeping it frozen and adding some new layers (for token or sequence classification for example) or by fine-tuning it.
The paper taken into consideration [3] leverages the capabilities of the previously mentioned PLM (Pretrained Language Models) of capturing cross-lingual word sense knowledge with Contextual Word-Level Translation (C-WLT), achieving WSD in a zero-shot setting.
More concretely, this work tries to disambiguate a target word w_s in a sentence (context) c_s where $w_s \in c_s$, written in a so defined 'source language' L_s, asking to a PLM to translating it in a different language L_t, called target language.
They authors defined through experiments a specific prompt to feed into the PLM, written in the source language L_s, for example English: 'In the sentence 'c_s', the word 'w_s' is translated into $L-t$ as '.
The output of the model is a vector of the probabilities of all the tokens in the vocabulary of the model to be the next token. More precisely there is a vector of probabilities for each token of the sentence (prompt) fed into the model, but for what concerns the aim of the task only the prediction after the computation of the last token of the sentence is relevant.

As shown in Figure 1, from all the predictions, filtered by inspecting all the possible translations obtained by a BabelNet query, the most promising/probable w_{top1} is selected.
Then from the top word w_{top1}, a set of possible senses are retrieved again from BabelNet and finally is checked whether there are some senses in the intersection of the source word w_s and the translated selected word w_{top1}: $s(w_s) \cap s(w_{top1})$, where s(x) is the possible senses set of a word x.
The paper also claims to extend the disambiguation method including multilingual translation, so for each target word w_s, translates it via PML in different languages and then takes the resulting senses with the most multiplicity across all translations.
Regarding the use of PLM, were chosen: BLOOM models and GPT-Neo. They come in several version differing in the number of parameters, they are generally very big with the smallest one consisting in 560 millions of parameters for BLOOM and 125 for GPT-Neo.
Results show that as the model size increases, the PLMs can encode more cross-lingual word sense

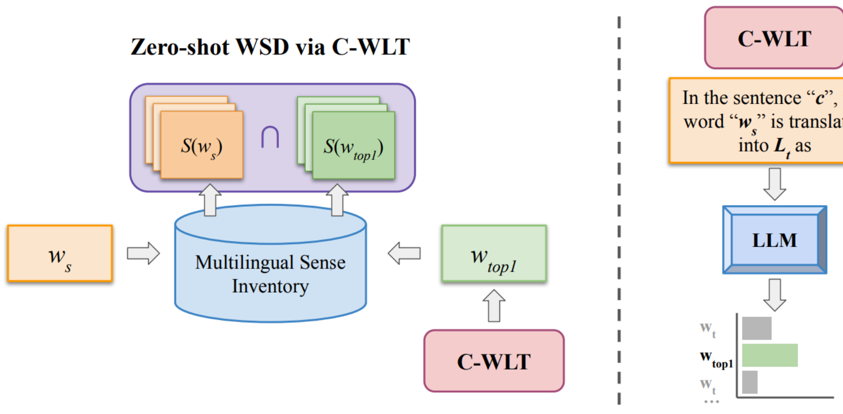

Figure 1: Overall structure of the paper, the right part shows the basic concept of the work i.e. the prompt to feed into the PLM and its outputs form. The left part shows the whole pipeline.

knowledge and better use context to improve Word Level Translation performance.
The dataset used is the XL-WSD dataset [5], which is comprised of 18 languages, and as multilingual word sense ontology to obtain translations and sense inventories BabelNet [4] as previously stated.

3 Project Details

This project is mainly a reimplementation of the paper work, but with some modifications.
The reference work on github is very messy and confusing, the code it's very poor of comments and somewhere even absent.
Furthermore it has no Readme.md file nor a description with the steps to replicate it, just files in alphabetical order with no interaction between themselves or with a clear hierarchy, it actually relies on saving data every time and reloading it, sometimes with no actual correspondence with the saving code and the actual files in the repositories.
The proposing code is more easily human-readable, rich of comments explaining what is done at that point. With also the possibility of not only evaluating the work with the dataset included, but also of running a single example: Given a sentence, a word to disambiguate and a selected model, it outputs the best translations from that model with its score and also the corresponding best senses in the intersection between the target word w_s and the top translation w_{top1} with their main glosses, a little textual description of the meaning of that word.

3.1 Workarounds

To be able to run large models like BLOOM or GPT a GPU is required to evaluate thousands of samples. In order to do so Google Colaboratory (Colab from now on) comes in help, but wiht the cost of stay strict to some of its implementation details.
This lead to a not easy use of Babelnet API as it requires Python 3.8, while the default Python version offered by Colab is currently 3.10, some workarounds can be made to install a new Python version, create an alias for the default exe, but the installation of new module is somewhat tricky and difficult. In few words, as stated in their github Google Colab doesn't support changes of Python version. More researches found that also Kaggle (a reliable alternative to Colab) doesn't.
Fortunately the paper repository comes with some data retrieved from Babelnet, with some dictionaries of (words : babelnet ids) or some with (dataset word id : translations).
Being that so, the best and fastest method was to remove the use of BabelNet in the work, for exception of the Example Run. In that modality the use of BabelNet is needed to allow the user to use as target word even words not present in the data retrieved by paper. Without BabelNet the Example Run should rely only on that previously mentioned data, making the Example Run just a filtering of

the Dataset Run for a given sample.

3.2 Method Modification

In addition, the behaviour of the translation phase has been modified.
A new version is proposed, but also the paper method is still present and it's possible to use it, but with a correction.

How the paper works:
After feeding the sentence prompt to the PLM, it retrieves the probabilities of the next token and also its cache, to avoid the recomputation of the prompt in the subsequent uses. Then, for each possible translation, it feeds again the model with all the candidate tokens one at a time, using the previously cached data and at the end of each word, it computes the average scores of all tokens.

How the paper should work:
Now, it seems that the actual intention of the paper's authors was to use a new cache generated from a token for the next token of a word, as the stored variable $model_cache$ in line 175 suggests.
But this variable remained unused, resulting in using every time the prompt cache for a new token prediction, even the last token, so the model would see as the new token of the sentence a mid-word token or even a final one, but it would interpret it as the starting token of a word, actually of the translating word.
Obviously this could generate some unwanted scores. So the modification has been to make use of that $model_cache$ variable, feeding each new token with the results of its predecessor. Of course for a new word, the starting cache used is always the prompt one.
But this could lead in some optimistic results, because with this behaviour the model predictions after the first one are biased: the model sees as the continuation of the prompt some tokens that can form a subset of the possible translations of that word, and so the scores will be lower for the possible resulting words and lowering the final score for that word, that might be a wrong translation, a wrong sense.
Take for example, not so probable but very clear to understand this issue, the word *'mozzarella'* as probable translation for the word 'cheese'. After the prompt we have a score for the first token of *'mozzarella'*, let's say the token that corresponds to *'moz'* that wouldn't be a so high score, but after then we feed the model with that token to predict the score of the next one along with the previous results, so the model will see that the sentence from which it will have to predict the next token is *"In the sentence ... the word 'cheese' is translated in Italian as '**moz**"*. Now the model is biased, and is more prone to complete the word, so the score of the next token of *'mozzarella'*, let's say *'za'* will be a lower one, maybe the most probable at that point.
Doing so, the final result will be an average of the more and more low scores of the word.

How the proposed variation works:
The new proposed method of inference tries to reduce at the minimum this model bias.
It first starts tokenizing at the same time all possible translations of the target word, then computes for each possible translation the shortest unique sequence of initial tokens, and stores that in a mask that will be used during the PLM query.
For example, if we have two possible translations '*tavola*' and '*tappetino*', that can have the same starting token corresponding to '*ta*', the proposed method will feed to the PLM the tokens corresponding to '*tavo*' and '*tapp*' respectively. Furthermore, if another possible translation is the word 'tappo', the resulting sequenecs will be '*tappet*' for tappetino and '*tappo*' for 'tappo'.
In Figure 2 is represented a little example of the selection method.
So, during PLM queries, if a possible traslation starts with a unique token, its score will be just the score of the original prompt, if not, it will be the average of the scores of its sequence, using the same logic of the paper.

Another little improvement introduced consists in modifying the prompt template that is used with the PLM, by just adding at the end of it a quote mark $'$.
This is justified having noted through experiments with different models that the most probable token after the original prompt was actually the quote mark, as the PLM models will predict/generate the

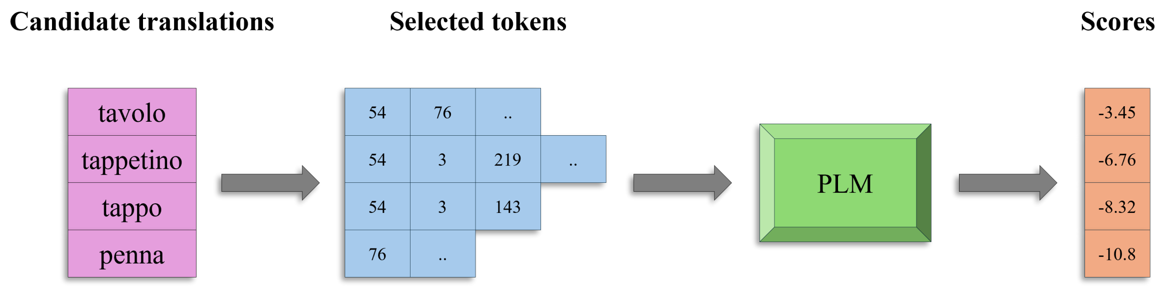

Figure 2: A sketch of the selection of the unique token sequences to feed into the PLM.

translated words through quote marks.
That being so, the addition in the prompt of the starting quote mark puts the output of the prompt to predict actually the first token of the translated word, making the probabilities taken into consideration more reliable.

4 Results

Regarding the actual experiments and evaluation of the project, the configuration used is the best-ensembled settings as the paper names it.
Is consists in using the prompt in English and as target languages English, Russian and Chinese.
Due to limited resources and time results are available with only the model BLOOM 3b, bloom model with 3 billions parameters, chosen in order to compare results with the reference paper, and the used source language is Italian, in order to better investigates errors and behaviours while coding.
As the metrics file states, the Recall is 61 and Jaccard Index is 51, which are a little less than the paper performances but still comparable. These little drops of performance might be explained with the strange behaviour of the paper inference while evaluating the tokens after the first one. But the method proposed in this project is more 'reasonable'.
Furthermore, actually the accuracy is still high, reaching 72, with 1635 correct disambiguation against 643 wrong ones.
To better understand the effect of the proposed project it would be better to evaluate it in other languages.

References

[1] Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. "Random Walks for Knowledge-Based Word Sense Disambiguation". In: vol. 40. 1. Cambridge, MA: MIT Press, Mar. 2014, pp. 57–84. DOI: 10.1162/COLI_a_00164. URL: https://aclanthology.org/J14-1003.

[2] Miller et al. "Introduction to WordNet: An On-line Lexical Database". In: *International Journal of Lexicography*. 1990. DOI: https://doi.org/10.1093/ijl/3.4.235.

[3] Haoqiang Kang, Terra Blevins, and Luke Zettlemoyer. "Translate to Disambiguate: Zero-shot Multilingual Word Sense Disambiguation with Pretrained Language Models". In: 2023. arXiv: 2304.13803 [cs.CL].

[4] Roberto Navigli and Simone Paolo Ponzetto. "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network". In: vol. 193. 2012, pp. 217–250. DOI: https://doi.org/10.1016/j.artint.2012.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0004370212000793.

[5] Tommaso Pasini, Alessandro Raganato, and Roberto Navigli. "XL-WSD: An Extra-Large and Cross-Lingual Evaluation Framework for Word Sense Disambiguation". In: vol. 35. 15. May 2021, pp. 13648–13656. DOI: 10.1609/aaai.v35i15.17609. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17609.

[6] Rocco Tripodi and Roberto Navigli. “Game Theory Meets Embeddings: a Unified Framework for Word Sense Disambiguation”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 88–99. DOI: 10.18653/v1/D19-1009. URL: https://aclanthology.org/D19-1009.