Comparing Formal and Distributional Meaning Representations: a Case of Information Retrieval

Giulio Cusenza, Giulio Posfortunati
University of Tübingen
giuliocusenza@gmail.com, giulio.posfortunati@gmail.com

Abstract

With the limits of current LLMs becoming more and more evident, formal meaning representations might still prove themselves useful for tasks in which reliable information plays a vital role. In this work we implement a simple tool which uses AMRs in order to answer a user's questions by extracting and reporting sentences from English Wikipedia articles. We then test this approach and compare it to an embedding-based system. We find that the major limitations of using AMRs in a real-world application are mainly speed constraints. Finally, we have a look forward at what further actions can be undertaken to improve such systems, and explore two partial solutions to increase speed.

Keywords: Meaning Representations, Information Retrieval, AMR, Smatch, SBERT

1. Introduction

The enthusiasm for formal representations of meaning seems to have diminished in the last years as large language models (LLMs) have started dominating most NLP benchmarks (Min et al., 2021). As the current limits of these models have become more evident, alternative solutions employing meaning representations might still prove themselves useful for tasks in which reliable information plays a vital role.

In this work we implement a simple tool which uses Abstract Meaning Representations (AMRs) (Banarescu et al., 2013) in order to answer a user's questions by extracting and reporting sentences from English Wikipedia[1] articles. We then test this approach and compare it to an embedding-based system using the Sentence-Bert (SBERT) model (Reimers and Gurevych, 2019). This comparison allows for an analysis of the advantages and challenges of the use of systems based on formal semantics over distributional semantics representations. We find that the major limitations of using AMRs in a real-world application are speed constraints due to high-complexity parsing algorithms. On the other hand, processing embeddings is remarkably quicker as it all boils down to vector operations. We suggest two strategies to partially solve the speed problem by pre-filtering the sentences from the Wikipedia article. Solutions to other problems (e.g., anaphora resolution) are also outlined. Finally, we draw some conclusions on what future research should focus on in order to further develop such tools.

2. The Application

For this study, an application was developed to explore how useful AMRs can be for referenced information retrieval, i.e. answering questions citing the source. The tool prompts the user to ask a question, and then returns a top list of sentences extracted from Wikipedia that could potentially answer the question. The exact inner workings of the system are displayed by the pipeline at Figure 1. In the following paragraphs we examine the single steps in detail.

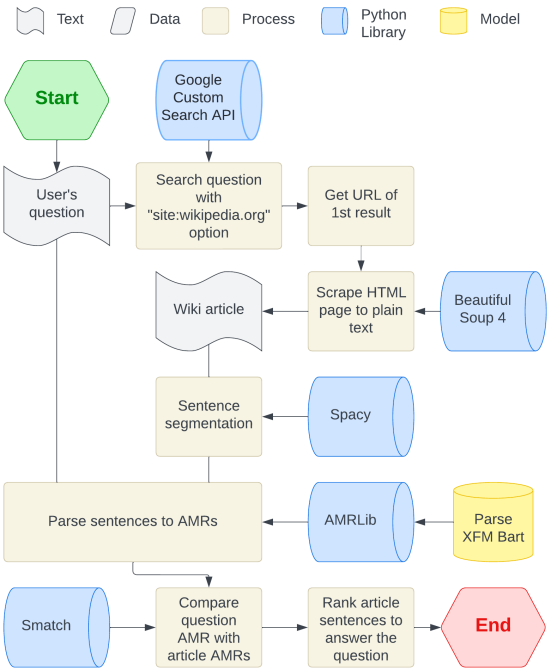

Figure 1: Pipeline flowchart.

The tool initially prompts the user to formulate a question. By design, the system is tailored to provide responses exclusively to inquiries necessitating factual information. Consequently, this precludes queries of a creative nature, those pertaining to personal preferences, or those demanding

[1] en.wikipedia.org

logical or mathematical inference, among others. Next, the question is searched through Google's Custom Search API with the scope set on the `en.wikipedia.org` website. In some cases, this API tends to answer the question itself. For example, the first search result to the question "Who was the first president of the United States?" will be the Wikipedia article about George Washington, which can raise doubts about whether this approach is directly affecting our results. While it is true that the Search API performs an important initial selection of the source, our system still has to base its answer on the content of the article. Moreover, it is not viable to run a semantic parser on a much wider selection of web pages due to the speed constraints of this technology (Section 3), therefore this step appears inevitable.

A further step uses the Beautiful Soup[2] Python library (version 4.12.3) paired with regex to perform a web scrape on the retrieved article. We extract the main text body and exclude all tables, citation indices, and IPA specifications, which affected the quality of the sentence segmentation system. Once the article has been extracted from the HTML file, it gets segmented into sentences through the spaCy pipeline `en_core_web_sm`. This model occasionally finds sentence boundaries where there are not supposed to be any, e.g. "lit." meaning "literally" caused a sentence to be mistakenly split into two. These problems are reflected further down the line, so a better sentence segmentation model would be a good improvement for the system.

The next step takes the user's question and the segmented sentences from the article and turns them into AMRs. This is achieved through the `parse_xfm_bart_base` model. This model takes a sentence and parses it into its corresponding AMR. The selection of this model over others is justified by its speed: with 31 sentences per second, it appears to be the fastest model, all with a marginal sacrifice in terms of accuracy.[3] Once the AMRs are available, it is possible to compare each sentence from the article to the user's question by measuring the Smatch score (Cai and Knight, 2013) between them. As the authors describe it, Smatch is "a metric that calculates the degree of overlap between two semantic feature structures". This means that the sentences can be sorted on the base of their Smatch score so to rank first those whose parsed AMR is most similar to the query's. This method allows us to create a ranked list of sentences where the top ones are most likely to be related to the user's question and possibly answer it. Some example queries and outputs can be seen at Figure 2.

[2] www.crummy.com/software/BeautifulSoup/

[3] github.com/bjascob/amrlib-models

```
Ask a question about a famous person, event, place, or anything:
> Who was the first president of the United States?
Ranking sentences from:
https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States
        1. He is the only U.S. president to have served more than two terms.
        2. The first president, George Washington, won a unanimous vote of the
           Electoral College.
        3. He was, and remains, the only U.S. president never affiliated with a
           political party.

Ask a question about a famous person, event, place, or anything:
> When was the telephone invented?
Ranking sentences from:
https://en.wikipedia.org/wiki/History_of_the_telephone
        1. Early telephones were technically diverse.
        2. Signaling began in an appropriately primitive manner.
        3. Their maximum range was very limited.
```

Figure 2: Two examples of command-line interactions with the tool.

3. Comparison

At the core of the tool described above is essentially a ranking system based on Smatch. We compared this system based on formal MRs with one built on distributional semantics. This second tool uses the Sentence-BERT model (Reimers and Gurevych, 2019) to generate sentence embeddings, which can then be compared on a vector space to find semantically neighbouring sentences.

The two systems – which we will refer to as Smatch and SBERT for shortness – were thus tasked with the ranking of answers for 30 different questions across 30 different Wikipedia articles. We annotated the first 20 sentences of each article labeling them as `2` (relevant), `1` (somewhat relevant), or `0` (not relevant). This annotation served as a gold standard to compute the Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2002), a measure of ranking performance spanning from 0 to 1, where 1 indicates a perfect match with the ideal order. The results of this analysis are reported at Figure 3. However, it must be noted that this task assumes that possible answers to the question will show close semantic similarity to it, while this is not necessarily the case. It should also be observed that, given the extremely limited size of the test data, these results are not to be taken as absolute measures of the systems' performance, but only as a comparison. Together with NDCG, we also measured the time required by each system to complete the ranking task for each article. These measures are visualised at Figure 4.

It is evident how the SBERT system is both more accurate and significantly faster. Its average NDCG is 0.80, compared to Smatch's 0.52, but the most significant gap is in the time performance: SBERT shows an average of 0.47 seconds per article, while the Smatch system about two minutes (118.09 seconds). This is due to the fundamental difference in the underlying operations performed by the two systems. For Smatch to work, the sentences have

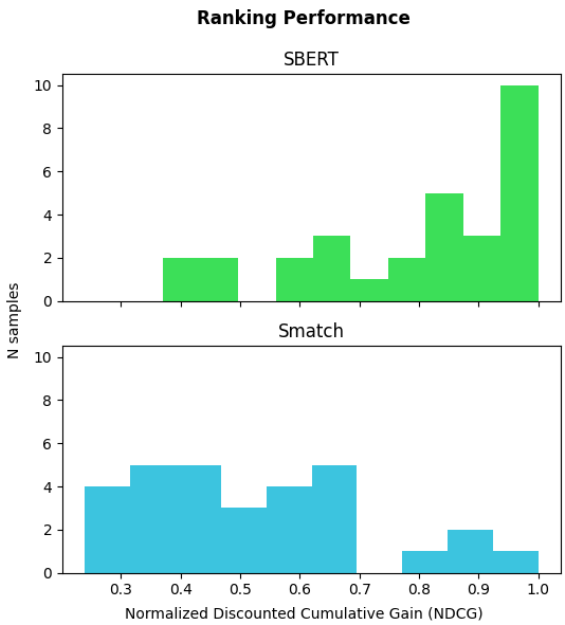

Figure 3: Distribution of the 30 Wikipedia articles based on the NDCG measures for each system.

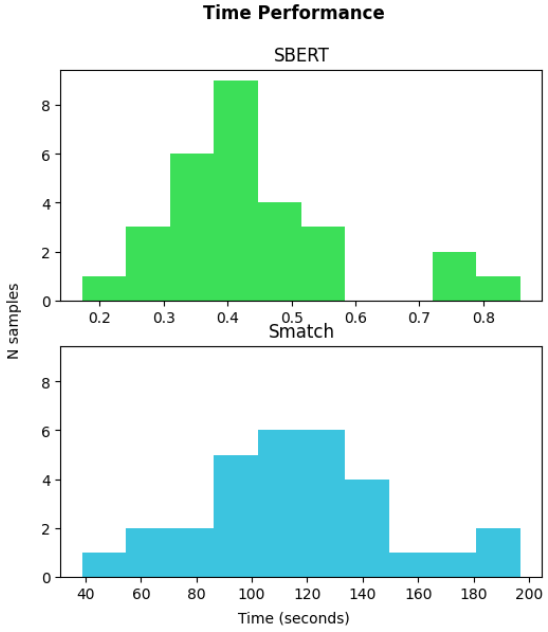

Figure 4: Distribution of the 30 Wikipedia articles based on the time each system required to rank the sentences (one outlier removed from the SBERT plot).

first to be parsed into AMRs. The parsing algorithm has a high asymptotic complexity, meaning that the number of operations required to parse a sentence grows very quickly as the sentence length increases. On the other hand, processing embeddings is remarkably quicker as it all boils down to vector operations, which computers perform extremely quickly.

4. Proposed Improvements

4.1. Skimming

On the base of the results of Section 3, we decided to concentrate our efforts on strategies to speed up the AMR-based system. The most basic one we designed involves a lemma-based skimming of the sentences: the main arguments of the question are identified through the AMR predicate structure and their lemmas saved in a set (e.g., "Who invented the telephone?" → $\{invent, telephone\}$). We then extract the lemmas of each sentences in the article using spaCy's `en_core_web_sm` pipeline. Next, we take the intersection of each sentence's set of lemmas with the question's set. All sentences where such intersection has a 0 cardinality are excluded. At the end, only the sentences containing any of the main lemmas of the question are kept for Smatch ranking.

4.2. Pre-ranking

A pre-ranking system was designed as an improvement on the basic skimming solution. Instead of excluding the sentences with cardinality 0, cardinality is used as a pre-ranking criterion, so that sentences with high cardinality (i.e., many shared lemmas with the question), are ranked higher. A k parameter is later used to determine how many sentences are taken from the top of the list for further Smatch-based ranking.

4.3. Limitations

Pre-filtering candidate sentences clearly aids in speed, but it has some important limitations. While effective for inflection, lemmatisation does not address other morphological phenomena such as derivation and compounding. Another limitation is represented by synonyms: a sentence answering the question does not necessarily include the same words. A solution to this could be offered by mapping words to PropBank[4] frames instead of lemmas, but this would pose the problems of named entities, which are not included in PropBank. A hybrid system, possibly paired with named entity recognition (NER) tools, could be devised as a solution to these problems.

5. Discussion

We primarily focused our attention on the possible alternatives, to the LLMs, that mainly rely on semantics and may result in a more reliable tool. The two approaches – Smatch and SBERT – are used as solution to the emerging issues deriving from

[4] propbank.github.io

the increasing use of LLMs, such as hallucination or exaggerated expression of certainty, as well as a lacking clarity on the information sources. A substantial difference between the two is the customisability of Smatch compared to SBERT. The former, given its graph structure, allows a further refinement of the strategy, while the latter, characterised by a more black-box nature, hinders this possibility. With SBERT, the query can only be processed altogether, while with AMRs specific elements of the graph can be utilized to obtain more detailed results. The low speed performances revealed to be the main challenge. Although the AMR parser documentation (see Footnote 3) claims a speed of 31 sentences per second, our empirical results suggest a significantly lower speed. We attribute this discrepancy to a supposed length difference between sentences on Wikipedia and those used in the parser evaluation. We tried different lemma-based solutions (Section 4) that considered the limitations, we see as promising approaches to this speed-performance. Anaphora resolution poses another significant challenge, given the fact that the text is broken down into single sentences unrelated to each other. In cases where sentence components are expressed by pronouns, the sentence is ranked lower than if the components were explicit (e.g., "He was elected as first president", when asking about who was the first president of the United States). We consider this challenge impossible to be addressed at the sentence level. Therefore, we believe it requires further investigations for alternative solutions. One of such solutions, that we have explored, is DocAMR (Naseem et al., 2022). A DocAMR is a multi-sentence AMR that consider a document level graph representation, which resulted from the merge of each document sentence graph representation. On the other end, the direct consequence of a DocAMR use is the difficulty of mapping the information of source sentence.

6. Conclusion

Considered the difficulties represented by the low speed performances and the anaphora resolution, it results hard to state whether a pure formal meaning representation approach can show great improvement in the near future, as poor speed performance constitutes a constraint to further and wider investigation on large texts. An interesting scenario would be to take into account hybrid systems where the best of both methods – formal and distributional – is used to accomplish such tasks. This hybrid systems could benefit from the speed performance of distributional methods, while making use of the fine-grained information that AMR graphs can provide. A trivial system could use embeddings for pre-filtering and AMRs for information extraction. Overall, we believe that our results are encouraging and represent a good motivation to inspire further research on formal MRs.

7. Bibliographical References

Tatiana Anikina, Alexander Koller, and Michael Roth. 2020. Predicting coreference in Abstract Meaning Representations. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 33–38, Barcelona, Spain (online). Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey. *CoRR*, abs/2111.01243.

Tahira Naseem, Austin Blodgett, Sadhana Kumaravel, Tim O'Gorman, Young-Suk Lee, Jeffrey Flanigan, Ramón Astudillo, Radu Florian, Salim Roukos, and Nathan Schneider. 2022. DocAMR: Multi-sentence AMR representation and evaluation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, pages 3496–3505, Seattle, United States. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.