Running Semantic Search Over Complete English Wikipedia on a Local Computer

Stojancho Tudjarski
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje, North Macedonia
stojancho.tudjarski@students.finki.ukim.mk

Ana Madevska Bogdanova
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje, North Macedonia
ana.madevska@finki.ukim.mk

***Abstract*—We implement a system that allows providing human-like answers to human-like questions extracted from a considerable amount of data in a reasonable time measured in seconds. To prove that the volume of the data used as a knowledge base where the answers to the questions are searched for, we used a complete English Wikipedia dump running on a local laptop under Windows10 OS, exposed to a software that receives questions and provides the three most relevant solutions. The entire technology stack of the implementation is the subject of this research.**

The main conclusion of this research is that it is possible to implement semantic search over a vast amount of text data on a local computer with an average hardware specifications, which is of outermost importance in developing different NLP systems.

***Keywords*—semantic search, deep learning, transformers, BERT**

I. Introduction

The semantic search is where the semantic meaning of the question is inlined with the provided search results. According to Fazainga et at.[1], there is no unique definition of the notion of semantic search on the Web. However, the most common use is the one as an improved form of search on the Web, where meaning and structure are extracted from both the user's Web search queries and different forms of Web content.

According to Berners-Lee at al.[2], The Semantic Web is a vision about an extension of the existing World Wide Web, that provides software programs with machine-interpretable metadata of the published information and data. In this way, the contextual meaning of the content of the web page would be embedded into the HTML with appropriate extensions, easily recognizable by a software. As a final result, this would expose them to an advanced search engines to be easily recognized as potential candidates for search queries where matching of the contexts of the question and the answers would play a significant role.

Therefore, it goes beyond looking for exact or similar matches of the words used in the search queries in the answers provided at the end of the search process. That leads us to the possibility of structuring questions less formally and more humanly, still recognizable by the software, which is capable of extracting answers that do not necessarily contain the exact words used in the questions, sill the meaning of the answers is with (at least partially) preserved correctness from a semantic perspective, as recognized by the humans.

Unfortunately, this ideas never gain wide adoption, due to the several factors. The most significant one is the fact that is hard to design a taxonomy of terms that would cover a wide area of topics that would be both detailed enough to cover all necessary subjects of the humans' interests, and to be acceptable for a wide population of naive users, i.e., ordinary end users who are not necessarily familiar with domain specific semantic data and just read news, and on the other side, highly specialized experts that look for very specific terms, context-wise.

The problem of the semantic search have been a subject of various research studies and applied with variable success using various approaches. Still, the breakthrough in this field is put on the scene by introducing the Transformers neural network architecture and with the BERT language model as the first helpful result of it [Attention is all you need], both applied by Google's researchers. In the following months, BERT became the fastest adopted new technology in the field, on such a scale as Google's search engine is.

The rest of the paper is organized as follows. Section 2 presents an overview on similar papers or related researches. The implemented methods, including the data set and the model selection, is described in Section 3. The results and the discussion are presented in Section 4. Section 5 presents the conclusion. The following text will provide technical details on how this technology can be applied on an average-performing home computer/laptop, still using the same technology running on Google's servers.

II. Related Work

Semantic search engine for XML documents is elaborated in the work of Cohen at al. [3], and the same approach can be used for HTML content as well.

The fact that searching by keyword and do not understand polysemy and synonymy are some reasons the traditional search engine is not suitable anymore, is well documented by Kasim at al.[4]. He successfully pinpoints the need of semantic search engine that is born of traditional search engine, but with a goal to overcome this limitation.

One implementation of semantic search is SemSearch engine, described by Lei at al.[5] that pays a special attention to hiding the complexity of semantic search from end users and making it easy to use and effective.

GeneView, described by Thomas at al.[6] is built upon a comprehensively annotated version of PubMed abstracts and openly available PubMed Central full texts. This semi-structured representation of biomedical texts enables a number of features extending classical search engines. For instance, users may search for entities using unique database identifiers

or they may rank documents by the number of specific mentions they contain. Annotation is performed by a multitude of state-of-the-art text-mining tools for recognizing mentions from 10 entity classes and for identifying protein–protein interactions.

A plausible approach to the semantic search problem is presented by Nagarajan at al.[7]. He presents an idea of giving semantic to a web page so a system can understand the semantic behind the web page which automatically increases the efficiency of information search.

Dietze at al.[8] introduces GoWeb, which combines classical keyword-based Web search with text-mining and ontologies to navigate large results sets and facilitate question answering.

The need of organizing data in easily findable, accessible, interoperable and reusable way, exposed to effective searches without exact text/word matching is recognized by Sadeeq at al.[9]. In his paper, he provides a wide review of Semantic Search and Semantic Web techniques. Various types of semantic search engines are investigated and the differences between the Semantic Search and Semantic Search keywords are determined. Additionally, the benefits of using Semantic Search were highlighted.

III. Implementation Methods

A. Datasets

We used a complete English Wikipedia dump, available on https://dumps.wikimedia.org. Various dump files are available:

- Database backup dumps, in a form of MySQL DB dumps
- Static HTML dumps
- Backup dumps of wikis which no longer exist
- Analytical data files
- Titles and descriptions only
- URLs to the pages only
- Complete articles, all the revisions included
- Complete articles, the last revision only

We used the last one, with complete articles included, but without the revisions history, so only the latest versions of the pages were used. The dump file is compressed in a format that allows parallelized decompressing, optimized for speed.

B. Data Cleanup

The data dump, decompressed, is in the format of one big XML file, 42 GB big. When XML tags are completely removed from it, still there is a lot of unusable text that acts like a noise to the search procedure, increasing the data volume, search time, and lowering the accuracy of the results and the quality of the provided data as the result.

, with many tags and metadata surrounding the leading articles that appear on the wikipedia.org site when some specific page is visited. For example, when an image is embedded in the page, the content contains the author and description of the image that is not directly connected to the page's content.

Some of the most valuable cleanup steps that did effective cleanup of the most of unwanted content are the following ones executed:

- Removing XML tags. What remains after this step is a content in Wiki syntax, that has to be cleaned further.
- Removing non-XML tags that provide information about the content in just a few words, but are not suitable for presenting to the user as a search result text.
- Removing the names of the main categories and sub-categories. Reason: again, does not provide plausible information to the user in a form of human-readable plain text.
- Removing all extra characters that act like a noise. Examples: '—', '*' and '='.

We cleaned up the XML tags metadata too short to contain any semantically valuable information. The cleanup was implemented with a Linux sh script containing 62 sed commands. This procedure took 22 hours to run on a Lenovo ThinkPad laptop with 16 GB of RAM and an Intel i7 processor, running Windows 10 as the primary OS but utilizing his Linux subsystem because of its fast 'sed' command.

C. Data Split

The texts were intended to be processed by the Hugging-Face Transformers library with the BERT (or his variation) fine-tuned with SQUAD dataset for the Q&A downstream task. One limitation of the BERT and similar models is that the input can be 512 tokens long. That means that we had to split the content of a big single-file Wikipedia dump into many small pieces of text files with content that does not exceed 512 tokens when tokenized. Otherwise, either we will produce error during the searches, or all of the tokens that exceeds 512 would be ignored.

In addition to that, we implemented text split with overlapping, meaning that the following text piece of the same article will contain the last 150 tokens of the previous text. The reason for this is the intent not to split texts in a way that possible answers to some questions would be split into two texts.

The side effect of this approach with a sliding window with padding is data redundancy. Still, since the final selection of the answer to a given question was made according to the biggest confidence score, this did not affect the user experience.

Also, a positive drawback of this padding is a fact that it is possible that a same answer to the same question might appear in two different texts. Since we are ranking the provided answers accordingly to the relevance score, and it is calculated within the provided single document only, during the question extraction phase, this approach give additional benefit of regularizing relevance score, because the bigger one will be chosen, pointing to the same answer, therefore, the chances of neglecting valuable but omitted search result due to accidental low relevance score with having contextual surrounding, will are effectively lower.

One technical detail on splitting text into chunks: the total number of files was too big to be stored in a single folder. Both Windows and Linux OSs showed problems handling folders with more than 60.000 files. We created subfolders in three levels to solve this issue, each containing no more than 1000 files. The complete text split duration: 19 hours.

D. Data Ingestion

Q&A pipeline consists of implementing a full-text search over provided texts, with the user question provided as an input. To perform fast searches over more than 19276500 files, we used Elastic Search. Its concept of inverted indexes showed its potential on a significant scale, having an average duration of the searches of less than a second. Since the texts were physically present on the disk in many small files, we used FSCrawler [https://fscrawler.readthedocs.io/en/fscrawler-2.9/]. Time spent on this task: 8.5 hours.

E. BERT Model Selection

We used BERT-like models available on the Huggingface models' repository, fine-tuned to Q&A downstream tasks. There were 45 models at that time, and we were doing this in April 2020. All of the models were fine-tuned using SQUAD 1.1 or 2 dataset. SQuAD stands for Staford Question and Answering Dataset. Original SQuAD 1.1 contains 107,785 question-answer pairs on 536 articles. The models trained with it showed SOTA results in finding the answers to the given questions in provided texts that should contain the answer. As a result, such models effectively answer any question in any provided text, no matter whether the text contains a valid answer to a question or not. These models were not trained to recognize an absence of a valid answer in a given text.

SQuAD 2.0 was introduced to overcome this drawback, with an additional 50.000 examples of questions and provided texts where no answer is labeled. As a result, BERT-like models trained with this dataset showed slightly lower accuracy in providing valid answers but can effectively recognize situations with no answer to a given question in a provided text.

Beyond this variation of the available modes, the following classifications were also considered:

Model size:

- small
- medium
- large

Model variation:

- original BERT
- RoBERTa
- AlBerta

There are combinations of models with all the possible sizes and variations, trained both with SQuAD 1.1 and SQuD 2.0 datasets, and in the end we ended with experimenting with 48 models

The bigger models showed better results in finding answers but were significantly slower, and that was unacceptable having the hardware we had in our possession when we were running the experiments. Therefore, we had to find the right balance with a reasonable accuracy vs. speed tradeoff. The resulting model was a small RoBERTa model that was fine-tuned with the SQuAD 2.0 dataset chosen among 45 applicable ones available on Huggingface.

IV. Results and Discussion

The semantic searches were enabled after performing all of the described data preprocessing and preparation steps. The interaction with the proposed Q&A engine is enabled by two different approaches: a slight application build using Streamlit Python library and a Flask-based website.

In Fig. 1 is presented one example of the Q&A engine in action. There might be (and usually there are) multiple answers to a certain question. Answers are ordered by the relevance score, in descending order, and the multiplicity of answers is expected when the relevance scores are close with their values.

One possibly confusing fact is that the relevance scores are calculated for the given question having only provided text. Consequently, the relevance scores are not necessarily comparable between two different texts, but the differences are not significant and reflect the human reasoning and the context of the question and the texts quite well. Normalizing texts, i.e., moving over long texts with overlapping sliding windows, as described above, makes this issue less recognizable.

V. Conclusion

In this paper we implemented a system that provides human-like answers to human-like questions. The knowledge base is extracted from a big amount of data in several seconds, using only a local PC/laptop hardware environment. The used data knowledge is the complete English Wikipedia dump. The total volume of data, after uncompressing and before data cleanup was at about 42 GB. The main conclusion of this research is that it is possible to implement semantic search over a vast volume of text, in a way that Google does, on a local computer with an average hardware specifications.

References

[1] B. Fazzinga and T. Lukasiewicz, "Semantic search on the web," *Semantic Web*, vol. 1, no. 1-2, pp. 89–96, 2010.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 34–43, 2001.

[3] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "Xsearch: A semantic search engine for xml," in *Proceedings 2003 VLDB Conference*. Elsevier, 2003, pp. 45–56.

[4] J. M. Kassim and M. Rahmany, "Introduction to semantic search engine," in *2009 International Conference on Electrical Engineering and Informatics*, vol. 2. IEEE, 2009, pp. 380–386.

[5] Y. Lei, V. Uren, and E. Motta, "Semsearch: A search engine for the semantic web," in *International conference on knowledge engineering and knowledge management*. Springer, 2006, pp. 238–245.

[6] P. Thomas, J. Starlinger, A. Vowinkel, S. Arzt, and U. Leser, "Geneview: a comprehensive semantic search engine for pubmed," *Nucleic acids research*, vol. 40, no. W1, pp. W585–W591, 2012.

[7] G. Nagarajan and K. Thyagharajan, "A machine learning technique for semantic search engine," *Procedia engineering*, vol. 38, pp. 2164–2171, 2012.

[8] H. Dietze and M. Schroeder, "Goweb: a semantic search engine for the life science web," *BMC bioinformatics*, vol. 10, no. 10, pp. 1–13, 2009.

[9] M. J. Sadeeq, S. R. Zeebaree *et al.*, "Semantic search engine optimisation (sseo) for dynamic websites: A review," *International Journal of Science and Business*, vol. 5, no. 3, pp. 148–158, 2021.

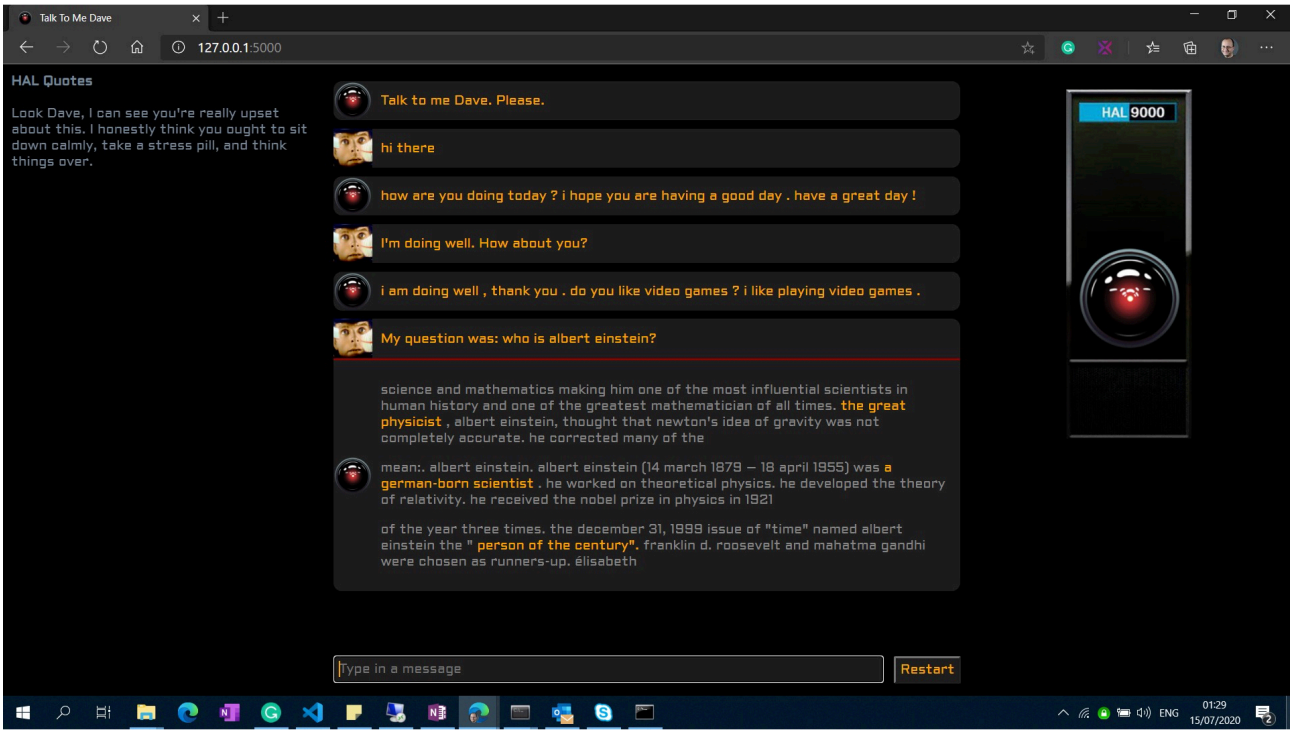

Fig. 1. Q&A in action