On My Disk integration: Milestone 2, Deliverable 4 Catering for multilinguality

The PeARS team members

This technical report describes the adaptation of the PeARS model that will be used by the On My Disk (OMD) system from English to other languages, particularly the OMD project languages French, Slovenian, and French. We compiled several multilingual datasets in order to evaluate performance and compare performance between languages. Initial experiments, using the same pipeline that had been shown to work well for English, showed that performance on the other languages lagged behind, possibly due to their higher morphological (grammatical) complexity. We addressed this issue by implementing a positional indexing system, which increases basic query matching performance on all languages (including English) to a near-perfect level.

1. Introduction

As a European project, multilinguality is at the very core of the collaboration project between PeARS and On My Disk, with a particular focus on French, Slovenian and Russian as well as English. Our previous work in the project has focused on adapting and evaluating the existing PeARS system – originally developed for English – for the purposes of On My Disk. In this report, we will report on our work that extends this work to the other project languages. In the process, we also introduce positional indexing, which very substantially improves keyword matching performance on all benchmark datasets, including the existing ones for English.

2. Data collection

In previous reports, we documented the evaluation of our system with English-language data. In order to simulate real-world use cases of the OMD system, we created several *personas*: datasets containing sample documents corresponding to several types of users (e.g., the 'HR specialist' persona is a dataset consisting of resumes, contracts, etc.). For performing multilingual evaluation and comparison, we extended this by creating two multilingual personas: "multilingual-news" (MN) and "multilingual-recipes" (MR). MN consists of the existing "news" persona for English, plus newly collected data for the other languages. MR was created from scratch for all languages.

2.1 Nature of the data

MN consists of 4000 recent news articles for each language. News articles could be representative of several realistic use cases (e.g. journalists, researchers, or ordinary citizens collecting clippings about topics of interest) but also have several other properties that make them suitable for use in search engine evaluation: they can cover a wide array of topics, and the fact that news articles often contain neologisms challenges the robustness of a system's word representations. For example, the word "COVID" did not exist until a few years ago, and a system trained on data before 2020 would not be able to recognize it. However, a robust search system should still be able to deal with such cases. Thus, the multilingual news dataset is a good test for our efforts to deal with 'unseen' words (see Section 3).

MR consists of 400 recipes and food blog posts for each language. Recipes represent a realistic use case of On My Disk – users might want to create an easily searchable collection of their favorite recipes. Moreover, recipes are a good domain for testing our system's robustness to different text genres: recipes might vary from short lists of ingredients and instructions (possibly with a lot of shorthand and abbreviations) to elaborate texts complete with inspirational background stories. Specifically, they provide a test case for our tokenizers and distributional semantic model that are trained on Wikipedia text: arguably, recipe blogs are further removed – in terms of vocabulary and word frequencies – encyclopaedic text than news reports are.

2.2 Collection process

We collected data from publically available news websites and food blogs via RSS feeds. By making use of feeds, we ensure that we have 1) recent data (perhaps not crucial, but a plus, considering the issue of neologisms discussed above); 2) a general way of collecting data from different sources without having to manually adapt to each website's structure; 3) easily extendable datasets: we can simply re-run our pipeline at any given time to add more articles to the datasets.

For each persona and each language, we first compiled a list of good sources. For MN, we first looked for the websites of national and regional public news broadcasting organizations,[1] and where necessary extended this with the sites of newspapers and major online commercial news outlets available in French, Russian and Slovenian, until we had a large enough number of news sources per language that would yield a sizable number of recent articles. For MR, we found suitable food blogs by querying commercial web search engines (DuckDuckGo and Google) for the translation equivalent of "food blog" and "recipes" in each language, restricting the website domain to `blogspot.com` and `wordpress.com`,[2] and adding the resulting URLs after manually inspecting them for suitablility.

After compiling a list of news sources for each language, we created an automated pipeline that uses existing Python libraries[3] that 1) check the sources in our list for new articles, parse the RSS feed and extract meta-data (e.g., headlines, URLs and publication dates) from it; 2) download the original web page and extract the body text from it. A limiting factor of RSS feeds is that they only contain the most recent entries (typically 5-20 articles/posts). Thus, in order to create sizeable datasets, a large number of feeds is needed and/or the scraping pipeline needs to be repeated until the desired number of entries has been reached. For copyright reasons, we cannot release the full dataset, but the scripts for recreating the dataset (with more recent articles) are fully open source.

1 Given the ongoing war in Ukraine and EU sanctions against some Russian state-owned media organizations, we opted to avoid including any news websites that are published in Russia itself. Instead, we relied on independent Russian-language media published outside of Russia (e.g. *Medusa*, *The Moscow Times*) as well as Russian-language news content published by western media organizations, e.g. *Euronews Russian* and *BBC Russian*.

2 This was done because Blogspot and Wordpress sites are guaranteed to have RSS feeds located under a predictable URL (e.g. main website url plus `/feed`), which made the collection process considerably more efficient. Additionally, Blogspot and Wordpress sites generally do not have pop-up advertisements or paywalls that would prevent scraping data from them.

3 These are "feedparser" (`https://pypi.org/project/feedparser/`) and "news-please" (`https://pypi.org/project/news-please/`).

persona	language	precision	recall	f-score	#returned	#gold
news	*en*	80	99	84	9.17	9.25
	fr	66	98	71	16.71	11.27
	ru	49	98	56	19.93	6.38
	sl	56	97	62	19.38	17.23
recipes	*en*	77	100	82	8.54	4.75
	fr	74	100	79	8.07	4.09
	ru	74	100	80	4.75	2.03
	sl	54	100	62	11.99	3.27

Table 1
Initial results. The columns '#returned' and '#gold' refer to the average number of documents that was expected viz. returned for each query. *: Results for *news/en* correspond to the English "news" persona reported on in the previous report.

3. Technical features & experiments

3.1 Tokenization

The first step towards implementing the PeARS search engine for documents in French, Slovenian and Russian is to train tokenizers that can split text into a fixed vocabulary of words and subwords. We used the same procedure that we previously implemented for English: downloading and extracting text from Wikipedia (public domain data that should cover a wide variety of vocabulary for each language) using the (in-house developed) Wikiloader package.[4] For each language, a recent 'dump' file was downloaded, raw text was extracted (i.e., excluding formatting, HTML code, etc.), and a subset containing 5 million words of text was created.

Next, we used Google's SentencePiece package[5] to train a byte-pair (BPE) tokenizer. We used hyper-parameter settings that we previously found to work best for English; in particular, we set the vocabulary size to 16,000, as a compromise between capturing as many complete words as possible[6] and reducing the computational and memory footprint of the system.

3.2 Initial evaluation

Results from the initial setup are shown in Table 1. Results for French and Russian are somewhat below those for English, and Slovenian is much lower. (??) It is not entirely clear what explains the big difference with Slovenian. One possible explanation is that the Slovenian tokenizer might be off less quality, given that the Slovenian Wikipedia is much smaller and perhaps with less diverse language compared to the English, French and Russian Wikipedias.

4 `https://github.com/possible-worlds-xyz/wikiloader`
5 `https://github.com/google/sentencepiece/`
6 BPE models work by counting the frequency of the words and parts of words (groups of characters) in the training corpus. Words that are frequent above a certain threshold – depending on the size of the vocabulary – will stored 'on their own' in the model (e.g. *the* $\rightarrow$ `_the`), whereas less frequent words are broken up into smaller pieces (e.g. *agoraphobia* $\rightarrow$ `_ag or a phobia`) that themselves are more frequent.

3.3 Positional indexing

In order to address the problem of unreliable performance across languages, and to further improve performance on English, we implemented a *positional indexing* (also known as *reverse indexing*) system. Such a system entails storing a dictionary for each document that maps each item in the tokenizer's vocabulary to a list of positions in the document where the items occur. Using such a system allows us to enforce exact keyword matches for words that are split into subwords. For example, if the word "watermelon" is split into `_water melon`, with our previous vector-based indexing approach there was no way to prevent the system from matching unrelated documents containing `_water` and `melon` separately (e.g. "I ate a melon by the waterfall") because only the frequency of each token in each document – and not their positions in the document — are taken into account. Instead, with positional indexing, it is possible to enforce any kind of desired constraint on the relative positions of subwords to each other. For example, it is possible to accept documents containing the subwords in the query in consecutive order (e.g. a document with `_water` at positions $\{110, 200\}$ and `melon` at positions $\{50, 201\}$ would match *watermelon* $\rightarrow$ `_water melon`) but a document with `_water` at $\{100, 200\}$ and `melon` at $\{300, 400\}$ would not. For now, we modified PeARS to rely exclusively on positional indexing, and use a scoring function that simply enforces exact keyword matches, but in the near future, we plan to integrate positional indexing with an improved version of vector-based indexing (taking into account semantic similarity between words and documents instead of only word counts), and to use a more sophisticated positional scoring function that takes into account both subword completeness and other factors such as giving higher scores to search terms occuring at the beginning of a document and to search terms occuring close to each other in a document.

We re-ran all of the experiments reported on above (Table 1) with positional indexing, and obtained F1 scores between 97-100 across all languages and datasets. This was exactly as expected since the positional indexing algorithm ensures virtually perfect keyword matching (with some minor exceptions, e.g. related to the encoding of unusual characters such as emojis). This comes at the cost of a substantially higher memory footprint compared to our previous vector-based indexing approach, up to between around 25%-70% of the size of the documents that are indexed. However, we see much room for reducing the positional index size using various compression techniques, and are to implement these in the very near future.

4. Conclusion

In this report, we reported on our efforts in creating multilingual evaluation datasets and adapting our system to French, Slovenian and Russian. The main improvement to our system that we made to this end was implementing a basic positional indexing system, which ensures virtually flawless keyword matching accuracy. This step has not only solved our previous tokenization-related challenges, but also provides the basis for future work on semantic search: we are currently already working on extending our vector-based indexing system to capture not just plain word counts of documents, but represent documents in such a way that allows for comparing and retrieving semantically related documents even if they contain different vocabulary items (e.g. retrieving documents about "puppies" when searching for "dogs"). Using a positional indexing system in combination with a (semantic) vector-based indexing will help to create a good balance between exact match accuracy and retrieving additional non-exact matching documents. In addition to work on semantic search, our short-term future work will focus on further improving the indexing systems and integration with the On My Disk API.