Extracting a domain-specific parallel corpus from Wikipedia

Iole Falsone, Alice Fedotova, Marianna Guarnieri

1. Introduction

1.1 Description of the problem

The aim of this project is to design and implement an efficient and semi-automatic pipeline for the identification of in-domain texts to be used in the creation of a comparable, or even better, parallel corpus. In the proposed scenario, our job is to provide the fictional company MyPoster with suitable data for training a domain-adapted machine translation (MT) system. In order to do so, a large amount of in-domain parallel or comparable texts is needed. Due to the wide range of clients and domains, it is crucial to automate the process of acquiring these texts to the highest extent possible. A common approach for creating parallel corpora is web crawling, which involves automatically accessing and downloading large amounts of text data from websites (Bañón et al., 2020; Morishita et al., 2020). While it is an efficient way to acquire parallel text, there are potential risks and limitations that come with web crawling, such as the possibility of obtaining copyrighted material or low-quality text. For this reason, MyPoster wants to avoid the use of web crawling. A potential solution comes from Wikipedia[1], the most popular source of multilingual copyright-free content on the Web. Due to its vast collection of articles and the fact that the same topic is covered in different languages, Wikipedia is an ideal resource for mining parallel texts in a large number of language pairs. For the purpose of this study, we explore the domain of video games in Italian and English.

1.2 Main findings and limitations of the approach

We propose an approach based on multilingual sentence embeddings to semi-automatically obtain parallel sentences from Wikipedia articles. We access Wikipedia through a Python wrapper library called Wikipedia API[2] and turn the in-domain bilingual texts extracted into bilingual sentence embeddings with Sentence-BERT[3]. Then, we compute the cosine similarity to match similar sentences: only those instances above the threshold are included to form the parallel corpus. This pipeline allows us to obtain a reasonably clean corpus of 10,069 texts fairly quickly. Furthermore, it can be easily applied to different domains with minimum human intervention and can support large quantities of data. Of course, searching the whole Wikipedia would be more effective for retrieving more diverse sentence pairs, but would also require further advanced computational resources. For this reason, we believe this approach to be a valid alternative.

[1] https://www.wikipedia.org/
[2] https://github.com/martin-majlis/Wikipedia-API
[3] https://www.sbert.net/

2. Conceptual solution

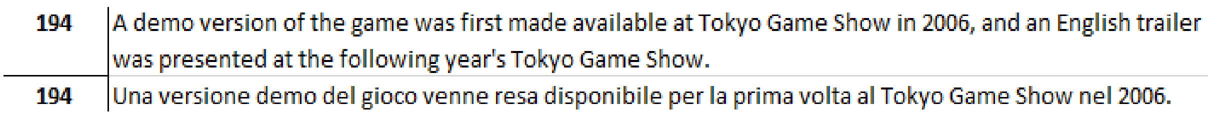
First, we need to estimate the amount of data required to efficiently perform this kind of study. According to the related literature (Park et al., 2022; Hasler et. al, 2021; Biçici, 2015), we established that to train a domain-adapted MT system, a suitable amount of data to start with would be around 10k parallel sentences. The next step would be finding the best way to retrieve such data. We experimented with the provided Wikipedia subsamples (1k, 10k, and 50k articles each) by identifying potentially relevant articles in English and in Italian. This could be easily achieved by applying some preprocessing to the corpus. We experimented with the 50k corpus. First, we excluded those text files shorter than 3 lines. After this first step, we were left with 27300 out of 50000 texts. Then, we tried to further clean the corpus excluding those files containing the words "Wikipedia", "File", "Thumb", "Thumbnail" or "REDIRECT/redirect" in the first paragraph. Finally, we used regular expressions (.*(v|V)ideo\s?game.*) to retrieve matches. This process left us with only 103 and 161 potentially in-domain English and Italian texts, which wouldn't be nearly enough to train a domain-adapted MT system. For this reason, we decided to explore the possibility of retrieving texts directly from Wikipedia. Drawing inspiration from Schwenk et al. (2019), we considered adopting what they call a local mining approach: since Wikipedia articles can have a corresponding page in other languages, it is possible to obtain parallel sentences by limiting the search to the respective articles. This is convenient both because Wikipedia articles usually have only a few hundred sentences each, making mining very fast, and because it is more plausible to find the translation of a sentence in the same Wikipedia article than anywhere else. Such an approach allows us to bypass the problem of identifying in-domain texts by directly using the desired category. It is also possible to apply this pipeline to other domains by simply extracting texts from the corresponding category (all Wikipedia categories here). Similarly to Schwenk et al., we also decided to take advantage of multilingual embeddings in order to extract parallel sentences from the comparable corpus that we obtained by retrieving articles about the same topics.

3. Solution implementation

The project was developed in Python 3[4] and is available in Jupyter format, which can be executed on the Google Colab platform[5].The main libraries used for the project are Wikipedia API and Sentence Transformers. According to Nurmanbetov (2020), distiluse-base-multilingual-cased is superior to LASER across 10 languages of the STS benchmark. For this reason, we decided to go for the former in order to obtain multilingual text embeddings. The model can be easily loaded using the Sentence Transformers library, which provides many state-of-the-art models that can be used to compute sentence embeddings. Wikipedia API is a Python wrapper library for accessing Wikipedia's API (Application Programming Interface). It supports extracting data from Wikipedia, including articles, summaries, and langlinks (links from the provided pages to other languages). We used Wikipedia API to retrieve the titles and the text of the articles contained in a given Category (e.g., Categoria:Videogiochi per genere) from the Italian Wikipedia. We assume that it is more likely to find a correspondent article in English than the other way around. We

[4] https://www.python.org/download/releases/3.0/
[5] The software and the related documentation are available at https://github.com/ffedox/pbr.

decided to extract the full text of each page, as we are interested in all potential parallel sentences contained within the articles. Then, we use the previously extracted titles to search for corresponding articles in English (this can be done using the langlinks property of Wikipedia-API). The result is a comparable corpus of texts dealing with the same topics in English and Italian. Articles that do not have an equivalent in the English Wikipedia are discarded. Once the comparable corpus has been obtained, we use NLTK's .sent_tokenize() method to split the texts into sentences with PunktSentenceTokenizer. We then obtain the multilingual sentence embeddings with *distiluse-base-multilingual-cased*, and compute the closest match for each sentence in the other language. The similarity between two documents is defined as the cosine of the angle between the two embeddings. We therefore use cosine similarity with a threshold of 0.8. If two sentences have a cosine similarity higher than 0.8, we include both of them in the final, parallel corpus. Once the parallel corpus has been extracted, we use the .drop_duplicates() method from the Pandas library in order to remove potential duplicate sentences from the parallel corpus. Additionally, we replace newlines with whitespaces and we remove all special characters except for punctuation and accented latin letters (many articles contained the original names of the video games in Japanese). To prevent short captions from being mistakenly identified as sentences, we also exclude sentences with fewer than six tokens. Lastly, we eliminate sentences with a significant token count difference (over 10). This is due to the fact that sometimes the model recognizes as parallel sentences that only partially overlap in content (Picture 1).



194	A demo version of the game was first made available at Tokyo Game Show in 2006, and an English trailer was presented at the following year's Tokyo Game Show.
194	Una versione demo del gioco venne resa disponibile per la prima volta al Tokyo Game Show nel 2006.

Picture 1: An instance where two only partially overlapping sentences were considered parallel by the model.

4. Outcome

By using the proposed method, we collected a total of 10,069 parallel sentences, which is slightly above the recommended amount that we found in the literature. Table 1 illustrates the size of the extracted corpus in more detail.

	Parallel sentences	Tokens (EN)	Tokens (IT)
Before preprocessing	11,920	317,078	329,676
After preprocessing	10,069	259,354	265,349

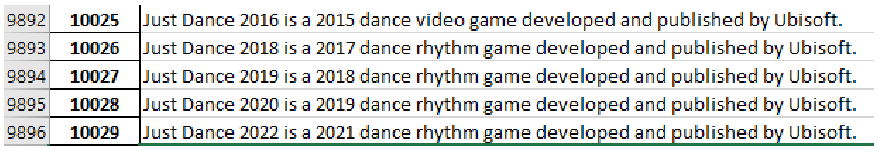
Table 1: Statistics about the obtained corpus.

We store the parallel sentences in a tabular format, exporting them to .XLSX. We also convert the Excel file into the .TMX (Translation Memory Exchange) format, as this is the one required by many CAT tools such as BootCat. In order to do this, we use a free online tool developed by Translatum[6].

[6] https://translatum.gr/cgi-bin/excel-to-tmx.pl

5. Discussion

The pipeline we implemented is fast and affordable. For instance, the time necessary to extract a parallel corpus from the “Avventure grafiche” category, which contains 396 articles in Italian (therefore 792 comparable texts), was estimated to be around 4 minutes and 50 seconds on a Tesla T4 GPU (the default one provided by Colab for free). Given that we are retrieving the full articles, we consider it to be a promising result. The extraction of the parallel sentences only took one minute thanks to the efficiency of SBERT. As a point of reference, starting from these 396 articles, we managed to obtain 1435 parallel sentences. Another advantage of this approach is that it can easily be applied to other domains as well by simply selecting the desired categories. For instance, one could visit the Contents/Categories page to identify a suitable category for the task at hand. It is also possible to use subcategories to explore more specialized domains. The extracted texts require only minimal preprocessing. Since we are downloading the full texts of the articles exclusively, unwanted files such as redirects or file description pages are excluded. In principle, one can obtain as many in-domain articles as there are on Wikipedia. However, it is possible to retrieve only up to 50 pages per API request. To overcome this problem, another possibility is downloading the whole Wikipedia (a 20 GB file for the English version) and searching it locally, provided that one has the computational resources to do it. This is what Schwenk et al. (2019) define as global mining, which consists in mining the entire Wikipedia for each language. The advantage of global mining is that the retrieved sentences are more diverse in structure. For instance, structures like “NAME was born on DATE in CITY” are more frequent when adopting a local mining approach. This is a problem because when this type of sentences is too frequent they are no longer of use to train an MT system. Picture 2 below is an example of this issue: the first sentence of the Wikipedia article remains unchanged, although the year of release of the video game is different per version. Therefore, Pandas’ .drop_duplicates() cannot identify these as duplicates.



9892	**10025**	Just Dance 2016 is a 2015 dance video game developed and published by Ubisoft.
9893	**10026**	Just Dance 2018 is a 2017 dance rhythm game developed and published by Ubisoft.
9894	**10027**	Just Dance 2019 is a 2018 dance rhythm game developed and published by Ubisoft.
9895	**10028**	Just Dance 2020 is a 2019 dance rhythm game developed and published by Ubisoft.
9896	**10029**	Just Dance 2022 is a 2021 dance rhythm game developed and published by Ubisoft.

Picture 2: Examples of almost identical sentences.

As already mentioned, the presence of too many almost identical sentences is not ideal for the training. In future work, we can address this issue by excluding the first sentence of each article. The obtained corpus can also be used to evaluate a domain-adapted MT system by using metrics such as BLEU or BERTScore.

References

Bañón, M., Chen, P., Haddow, B., Heafield, K., Hoang, H., Esplà-Gomis, M., ... & Zaragoza, J. (2020, July). ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 4555-4567).

Biçici, E. (2015). Domain adaptation for machine translation with instance selection. *The Prague Bulletin of Mathematical Linguistics*, *103*(1), 5.

Hasler, E., Domhan, T., Trenous, J., Tran, K. M., Byrne, B., & Hieber, F. (2021, November). Improving the quality trade-off for neural machine translation multi-domain adaptation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 8470-8477).

Morishita, M., Suzuki, J., & Nagata, M. (2019). JParaCrawl: A large scale web-based English-Japanese parallel corpus. *arXiv preprint arXiv:1911.10668*.

Nurmanbetov, D. (2020, May 31). *BERT Model Embeddings aren't as good as you think.* Towards Data Science.
https://towardsdatascience.com/cutting-edge-bert-nlp-model-bb0bfc8b7aec

Park, C., Kim, H., Calapodescu, I., Cho, H., & Nikoulina, V. (2022). Dalc: Domain adaptation learning curve prediction for neural machine translation. *arXiv preprint arXiv:2204.09259*.

Schwenk, H., Chaudhary, V., Sun, S., Gong, H., & Guzmán, F. (2019). Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. *arXiv preprint arXiv:1907.05791*.