BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

CROSS LINGUAL NEWS ARTICLE CLASSIFICATION AND AUTOMATIC TOPIC DISCOVERY USING MULTILINGUAL LANGUAGE MODELS

MEZI-JAZYČNÁ KLASIFIKACE NOVINOVÝCH ČLÁNKŮ A AUTOMATICKÉ OBJEVOVÁNÍ TÉMAT POMOCÍ VÍCEJAZYČNÝCH JAZYKOVÝCH MODELŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. ANETA DUFKOVÁ**
AUTOR PRÁCE

SUPERVISOR — **SANTOSH KESIRAJU, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

148255

Institut:	Department of Computer Graphics and Multimedia (UPGM)
Student:	**Dufková Aneta, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Machine Learning
Title:	**Cross Lingual News Article Classification and Automatic Topic Discovery Using Multilingual Language Models**
Category:	Speech and Natural Language Processing
Academic year:	2022/23

Assignment:

1. Scrape one or more news websites (who publish articles under CreativeCommons license, eg: global-voices, boomlive, ruralindiaonline), and build a multilingual dataset. Use the tags as ground-truth categories, where each article can belong to one or more categories.
2. Split the dataset into training, validation and test sets based on their chronology. For example, training and validation can be from 2000-2016, whereas test data can be from 2016-2022.
3. Benchmark existing pre-trained language models such as LaBSE, sentence-transformers, and LASER on the newly built dataset, where the downstream classifier can be simple binary classifiers like logistic regression or SVM or MLP.
4. Use clustering based techniques to identify similar articles across languages and represent the cluster with "most representative words" (topic discovery).
5. Develop a web-application to visualize / categorize / cluster news articles across various languages.

Literature:

- Feng et al, "Language agnostic BERT setence embeddings". ACL 2022.
- Artetxe and Schwenk "Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond". TACL 2019.
- Reimers and Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". ACL 2019.

Requirements for the semestral defence:

- Points 1, 2 and 3

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Kesiraju Santosh, Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	17.5.2023
Approval date:	31.10.2022

Abstract

The goal of this thesis is to perform cross-lingual classification and automatic topic discovery of news articles using pre-trained multilingual language models. For this task, no large multilingual dataset is available, so the first contribution of this thesis is to create one. The other aim of this thesis is to benchmark multilingual embedding models LaBSE and LASER2 in a classification task. This is done through various experiments, such as training on a limited number of articles and naturally zero-shot learning. Then, a topic discovery is performed so that an article can be represented not only by categories but also by the most representative words. Lastly, the results of classification and topic discovery are visualized in a simple web application.

Abstrakt

Cílem této diplomové práce je provést mezijazykovou klasifikaci a automatickou detekci témat novinových článků s využitím předtrénovaných multijazykových modelů. Jelikož pro tento úkol nebyla k dispozici žádná vhodná datová sada, prvním přínosem této práce je vůbec takovou sadu vytvořit. Dalším krokem práce je porovnat multijazykové modely LaBSE a LASER2 v úloze klasifikace. K tomu je využita řada experiment zaměřených na trénování na omezeném počtu článků a samozřejmě testování na jazycích, které nebyly použity při tréninku. Poté je provedena automatická detekce témat, takže článek může být reprezentován nejen kategoriemi, ale také odpovídajícími slovy. Na závěr jsou výsledky popsaného procesu vizualizovány v podobě webové aplikace.

Keywords

Natural Language Processing, LaBSE, LASER, multilingual classification, topic discovery

Klíčová slova

zpracování přirozeného jazyka, LaBSE, LASER, multijazyčná klasifikace, objevování témat

Reference

DUFKOVÁ, Aneta. *Cross Lingual News Article Classification and Automatic Topic Discovery Using Multilingual Language Models.* Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Santosh Kesiraju, Ph.D.

Rozšířený abstrakt

Cílem této práce je vytvořit vícejazyčnou datovou sadu a použít ji pro srovnání existujících předtrénovaných vícejazyčných modelů. Žádná podobná datová sada doposud nebyla zveřejněna, první úkol tedy spočívá v nalezení vhodných zdrojů novinových článků, které budou přeložené do mnoha jazyků. Z diskutovaných variant je vybrán web GlobalVoices.org, z nějž je získána datová sada obsahující téměř 250 000 článků v desítkách jazyků. Využívá se existující knihovny News-please, ta ale má své chyby, což vede například ke špatnému kódovému označení jazyků. Navíc neumí extrahovat všechna data, jež jsou v práci potřeba, zbytek je tedy získáván manuálně.

Celá datová sada je uložena ve složkách podle jazyků, každý článek je dostupný ve formátu JSON, kde jsou informace strukturované, a také v původní HTML podobě, aby bylo možné případně dohledat více informací. Meta informace o článcích jsou uloženy v samostatných souborech, v CSV souboru se nachází zařazení článků do kategorií, v JSON souboru jsou spárované překlady jednotlivých článků. Získané články jsou dále filtrovány, některé kategorie jsou sloučeny, což vede ke vzniku výsledných 25 kategorií finální datové sady. Ta obsahuje 17 jazyků, zvoleny jsou jazyky s více než 2 000 články.

Práce využívá dvou více-jazyčných modelů, LaBSE a LASER. Původním záměrem bylo otestovat nejnovější LASER3, ale ten pouze přidává podporu nových málo rozšířených jazyků. Technicky vzato je tedy používán LASER2. Oba dva modely jsou využity pro vytvoření embeddingů z jednotlivých článků, přičemž jsou adaptovány různé strategie. Jako nejúspěšnější se ukazuje technika vytvořit embedding z každého odstavce článku a článek pak reprezentovat jako průměr těchto embeddingů. Cílem více-jazyčných modelů je umožnit trénink na jazycích, pro které je dostupný dostatek dat, a používat ho pro méně rozšířené jazyky. S ohledem na tento princip je provedena řada klasifikačních experimentů, kdy jako klasifikátor jsou využívány MLP, SVM a logistická regrese. Jelikož jeden článek může patřit do několika kategorií, při klasifikace je využíván přístup One vs Rest. MLP dává mírně lepší výsledky, a proto je základem všech následujících experimentů. Rozdíl ve výsledcích experimentů používajících embeddingy vytvořené pomocí LaBSE a LASER je zanedbatelný, oba modely si vedou velmi dobře, a to dokonce i v úkolech, kdy je k tréninku použito omezené množství dat (2 000 článků). Výrazné rozdíly mezi výsledky pro jednotlivé jazyky nejsou, z čehož plyne, že modely jsou schopné velmi dobře zakódovat i méně rozšířené jazyky. Klasifikační experimenty jsou vyhodnocovány metrikou váženého F1 skóre, které se pohybuje zhruba mezi 0,82 a 0,89. Makro F1 skóre ukazuje o něco nižší hodnoty, což napovídá špatným výsledkům pro některé z 25 kategorií. Vysvětlit to lze tím, že určité kategorie nemusí být tak jasně definované, nebo je v nich zařazeno nižší množství článků. Potěšujícím zjištěním každopádně je, že skóre se neliší mezi jazyky, to znamená, že klasifikátor trénovaný například na řeckých textech, si vede velmi dobře v klasifikaci španělských textů apod. Tohle chování je od více-jazyčných modelů vytvářejících embeddingy očekávané a tato práce ho potvrzuje.

Již připravené embeddingy reprezentující jednotlivé články jsou dále využity pro úkol automatického objevování témat. Prvotním cílem bylo shlukování pomocí K-Means a reprezentace každého shluku deseti slovy získanými pomocí TF-IDF. Každý nový článek by pak byl zařazen do shluku a představovala by ho slova z daného shluku. Tento přístup se nakonec neukázal jako vhodný a místo něj se využívá více-jazyčný model, který se z Bag of Words reprezentace a embeddingů naučí matice embeddingů slov pro všechny jazyky. U nově příchozího článku se získá embedding, který se vynásobí s maticí naučenou modelem a použijí se slova s nejvyšším skórem.

Pro demonstraci výsledků je nakonec vytvořená jednoduchá webová aplikace pomocí Gradio hostovaná na Hugging Face Spaces. Aplikace je k nalezení na webu WWW. Umožňuje uživateli zadat text článku. Tento text je pomocí LaBSE zakódován do embeddingu, klasifikátorem zařazen do kategorií a modelem na objevování témat mu jsou přiřazena nejvíce reprezentativní slova. Webová aplikace si vede dobře i u článků v jazycích, které nebyly přítomné v datasetu vytvořeném z Global Voices, například v češtině. To opět ukazuje na kvalitu embeddingů vytvářených modelem LaBSE.

Cross Lingual News Article Classification and Automatic Topic Discovery Using Multilingual Language Models

Declaration

I hereby declare that this Masters's thesis was prepared as an original work by the author under the supervision of Santosh Kesiraju, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Aneta Dufková
May 14, 2023

Acknowledgements

I am deeply grateful for the invaluable guidance and support provided by my supervisor, Santosh Kesiraju, Ph.D. His expertise, insight, and feedback played an instrumental role in shaping my work and ensuring its success.

ధన్యవాదాలు.

Contents

Chapter 1

Introduction

There are thousands of languages spoken today in the world. The most talented linguists can speak tens of them. The current state-of-the-art multilingual models support hundreds of them. This thesis benchmarks two recent multilingual models on a classification task and then performs clustering. Until this moment, there is no existing dataset covering a higher number of languages and wide range of topics that would be suitable for this task. The motivation of this thesis is therefore to create such a dataset and do an extensive evaluation of multilingual models.

The first output of the thesis is, therefore, the dataset itself. The initial dataset consists of 245,821 articles in 51 different languages. It does not contain only information about the category classification performed in this thesis but also a lot of additional information, so it can be used for further work. The other contribution of this thesis is the comparison of embedding models, which was conducted on the created dataset.

Individual news articles are obtained from the multilingual news website GlobalVoices.org. Chapter 2 describes the choice of a news website, the process of web scraping using existing libraries, and additional work that had to be done because of missing features in the libraries. It also explains how such a complex dataset is stored and how the meta-information is saved. The obtained data has to be preprocessed. The first task is to take a look at the data and consider which categories have enough articles, which can be merged, and how many languages can be preserved. Some languages with insufficient data had to be omitted. Additionally, this chapter talks about the rules for a good train/test split for this task.

The introduction to embeddings and multilingual sentence embedding models is covered in Chapter 3. Then, important language models that are used in this thesis are presented.

After that, the techniques for obtaining an embedding representation of a document are discussed in Chapter 4. Experiments are made with embeddings created by three different strategies.

Chapter 5 is about cross-lingual category classification using embeddings created by different models. Various classifiers are used and many interesting experiments are performed, such as training on a low number of articles and zero-shot learning. This chapter compares the results of the models and proposes explanations for them.

Similarly, Chapter 6 focuses on the second task, topic discovery, and discusses different approaches. One of them could not be used in the end, and the chapter explains why.

Both the previous tasks were visualized as simple web application, so the result of this thesis is publicly available.

Finally, the key takeaways are emphasized.

Chapter 2

Obtaining Dataset

Every machine learning related task starts with the process of finding a suitable dataset. The modern world is full of data; however, obtaining it in a structured and clean form is not always an easy task. In addition, finding a dataset for supervised learning is complicated, especially if the aim is to perform a specific analysis.

News article classification may not seem like a very specific duty, but searching for a cross-lingual dataset suitable for category classification is a tedious activity with uncertain results. There are a lot of article datasets without categories, although these are not helpful. This is why one of the aims of this thesis is to create a cross-lingual dataset of articles and their categories. No existing multilabel dataset was diverse enough. There exists, for example, Reuters Corpora [10] but it only has 4 categories and less than 10 languages. Then there is a Multilingual Open Text corpus [16] containing 44 languages but no categories.

This chapter starts with a comparison of multilingual news websites and scraping the articles. After webscraping, the dataset is still in raw format, which is not applicable to the following utilization. The main task consists of determining which languages and categories should be omitted (in the case of languages) or merged (in the case of categories). This decision directly affects the results of the following tasks; hence, analysis of counts and correlations is necessary.

After this step, the dataset is almost ready. The text is, thanks to the News-please library, almost clean; just some minor improvements have to be made. There are tweets inside the main text of articles that are not translated; they are in a different language than the language of the article, so they have to be filtered out manually. The News-please library is not able to do it. Then, in the end, the dataset is split into train and test parts.

2.1 Multilingual News Websites

For obtaining articles in different languages, choosing a multilingual news website was needed. Multilingual means not two or three languages but tens of languages. Data extraction from tens of local news websites would be too demanding, not only because of the need to find those websites but also because of the different HTML structures of each website. Another Retrieving the information from separate websites like this would be a lot of inefficient work.

For this purpose, the focus has been on news websites that publish the news in various languages. This can mean both miscellaneous articles in various languages and the same

articles translated into more languages. The goal was to choose a website that contains articles in numerous languages.

Since the aim of this work is to create a dataset that will be available for further usage, the license needed to be taken into account. A Creative Commons license is desirable for later use of the dataset.

Given the above criteria, the GlobalVoices.org[1] website was selected to obtain the data. Other considered websites were TheConversation.com, news.mongabay.com, boomlive, ruralindiaonline, and a few more. These were left as option B if Global Voices doesn't provide enough data.

Global Voices is more of a community of international writers than a news website. It publishes not only news but also stories and opinions that are rarely seen in mainstream media. The stories are available in dozens of languages; therefore, one article can be accessed in various languages. All content is published under a Creative Commons Attribution-Only license, which means that anyone can share and adapt the material for any purpose, even a commercial one. The necessary condition is to give appropriate credit; hence, in the dataset, all the information, including the authors and URL, is preserved.

GlobalVoices.org has dozens of language mutations. After a brief investigation, it was discovered that probably all the articles are translated into English. For that reason, scraping was performed on the English version of the website. Other language mutations were kept aside for further examination in case the number of scraped articles was not enough. In any case, the assumption was that all the translations of the articles would be scraped together with the English versions of the articles.

Every language mutation on GlobalVoices.org is structured into categories. Although at first sight they seem the same, they are partially inconsistent. Even the number of categories differs between languages. The English, Czech, Italian, and Spanish versions of GlobalVoices.org were examined, and small inconsistencies were found. Some languages have more categories that don't exist in the English version.

A similar trend can also be observed after opening the particular article. At the bottom of the page, every article has its own tags. They are divided into two groups: regions and categories. The first one expresses the local affiliation of the article; the later one says to which categories the article belongs. It can be seen that one article can belong to many of the above-mentioned categories. There are sometimes inconsistencies in categories between language mutations in the article; some are missing and others are extra. For the purpose of unification, categories and regions in the English version of the article are considered ground truths.

2.2 Web Scraping Tools

The technique used for extracting data from a webpage is called web scraping. This is a general term for the practice of gathering data using different procedures rather than a program that interacts with an API. The term is mostly used for describing a process when a program queries a web server, requests data, and parses it to extract the requested information [13].

Web scraping tasks get more difficult if the content of the webpage is protected by login or some other mechanism limiting access. Fortunately, with Global Voices, this is not the case. The HTML source code with the text of the article is easily obtainable.

[1] https://globalvoices.org

The task of retrieving the dataset and its categories can be divided into two subtasks:

1. Crawling the news website and getting HTML source codes for individual articles.
2. Parsing the source codes, obtaining the texts of articles and their categories.

Various Python libraries can be used for crawling and scraping websites, and for the purpose of this thesis, two of them were examined. Newspaper3k can only scrape recent articles, which is not sufficient for building datasets of tens of thousands of samples. News-please [7] is more powerful; it also contains a crawler and is able to download articles from the desired period.

News-please library allows to scrape articles from a given webpage in a given time period. For every webpage, it saves .html and .json files. HTML file is the original source code of the site, and the JSON file stores JSON object with extracted information. By default, News-please extracts a lot of information, including title, URL address, authors, date of publication, and download date. It does not extract categories; therefore, additional work was required.

2.3 Categories and Regions Extraction

Since the goal was to build a dataset that would be useful not only for the particular task but also for future projects, there was a need to collect as much information as possible. This way, the dataset will be general and suitable for different tasks. For this reason, categories and regions were extracted from the source code thanks to the library BeautifulSoup [21].

The following approach was adapted: The whole English website, Global Voices, is scraped category by category. Web crawling is done manually, page by page, in each category. The link to every individual article is tested to see if it is unique in order not to download some articles twice. Then it is given to the News-please library, which saves the article in HTML and JSON formats. Both of them are kept for further use, mainly for the need to get some additional or clarifying information. The manually extracted information about regions and categories is added to a JSON file. Then all the translations of the article are downloaded the same way; the only difference is that translation links are not treated because they were already taken from the English version of the article. After this, the next URL of the English article is processed. The whole approach is illustrated in Figure 2.1.

During the scrapping process, inconsistencies in languages were detected. The News-please library, among other things, saves also language codes. For some reason, the language code extracted by the library is not always true. This inconsistency occurred mainly for the Yoruba language with the abbreviation “yo”, News-please library tells “up”, and for Vietnamese (“vi”), News-please tells “sq”. It was solved by replacing the wrong language code with the right one in meta files and preventing other mistakes of this type by extracting the language code directly from HTML code, not relying on the News-please library in this.

2.4 Meta Files

Some extra information will be needed for preprocessing and using the dataset. Mainly, it is crucial to know which articles are translations of other language variants. It would be possible to store it inside each JSON file representing one article, but this approach has a huge disadvantage. To read the information, all the JSON files would have to be opened,

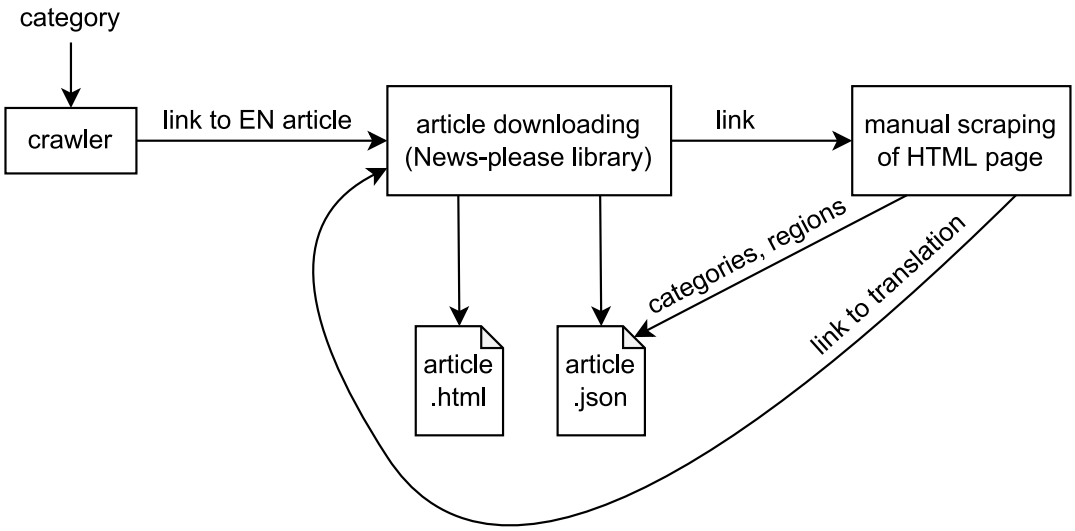

Figure 2.1: Articles are scraped by category. For each category, a number of pages is found, and then all links from the page are gathered. Each link is processed by the News-please library, which downloads .html and .json versions of individual articles. Manually scraped information is added to .json. Manual scraping is also used to get links to all translations of the article; these links are then sent to the News-please library to process them.

which is computationally ineffective. For easier processing and to make some statistics, plots, and decisions based on them, two files with meta information were created.

The first one is a CSV file, mainly with information about categories. It contains columns name (the name convention is described in the next section), URL address, date of publication of the article, and main category, which means a category under which the article was scraped (as mentioned above, the whole website was scraped category by category). Then there are columns for each category (e.g. Breaking News, Language, War & Conflict). Each row of this file represents one article, while for categories, there is always 0 or 1, depending on whether the article belongs to a particular category or not. This file was principally used to make decisions about merging some categories and omitting some languages. The file was also used to check for duplicity, where each article is identified by its unique URL address.

There are 45 categories in the English version of Global Voices, a subset is displayed in Figure 2.2. Some of them are speaking about similar topics; they can be merged into one larger supercategory. Some, like Breaking News, can be totally deleted since they don't have any specific meaning. To make these decisions, the number of articles in given categories and the correlation of categories have to be taken into account. This can be better observed from a chart or a correlation matrix.

The first visualization is illustrated in Figure 2.3. It is a small correlation matrix that shows how many articles are common for each pair of categories. Only nine first categories are displayed in view of the fact that, for all categories, the matrix is very huge. Numbers on the diagonal represent the count of articles in a category. It is important to remember that one article can belong to more categories, and in almost every case it does. It can be included, for example, in *Digital Activism* and *Censorship*, as these two go together, and *Citizen Media* at the same time. The *Citizen Media* category is the most numerous one, although very correlated with a lot of others categories.

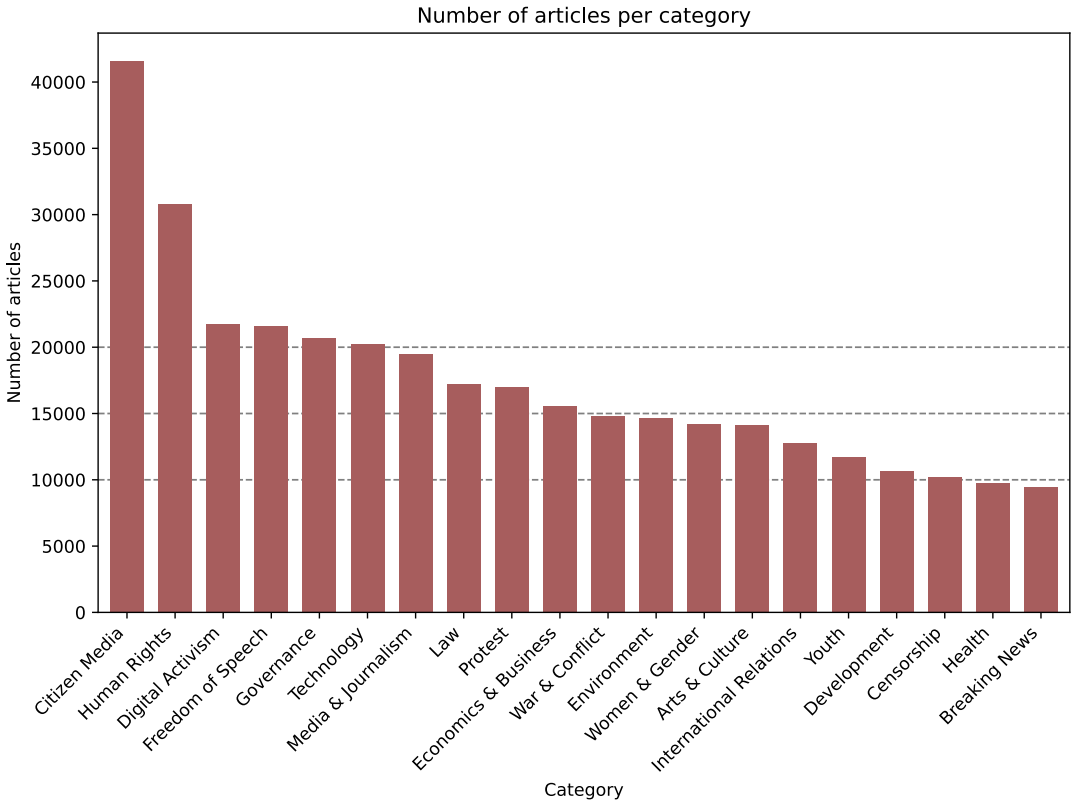

Figure 2.2: The image shows the number of downloaded articles per category. Only twenty of the most numerous categories are displayed. The least numerous were Photography with 845 articles, Language with 1324 articles, and Science with 2217 articles.

Other correlations can be better seen in Figure 2.4. It displays just a heatmap without article counts, where a lighter color means a higher number (stronger correlation). This symmetrix correlation matrix shows some clusters, groups of categories that often occur together.

Observations obtained from the above-mentioned charts are obvious. It was predictable that the same articles would belong, for example, to *Freedom of Speech* and *Human Rights* or to *Women & Gender* and *LGBTQ+*. Numbers validated intuition. Consequently, after studying charts and considering the number of articles in categories, new categories were created as follows: *Science + Technology*, *Women & Gender + LGBTQ+ + Youth*, *Literature + Arts & culture*, *Freedom of Speech + Human Rights*. *Breaking News*, *Good News* and *Citizen Media* were omitted. They don't have any specific meaning. Above that, almost all articles included in *Citizen Media* belong to other category or categories. Therefore, by deleting the *Citizen Media* category, no articles are deleted. *Humor*, *Ideas*, *Language*, *Film*, *Photography*, *Refugees*, *Music*, *Labor*, *Indigenous*, *Elections*, *Ethnicity & Race* and *Food* were omitted due to the low number of articles. The other categories remained as they used to be. To merge the categories, no changes were made in the files or file structure. Files are organized language-wise. All the changes were written only to the metafile. Original categories remained in .json and .html files; the dataset itself was not modified.

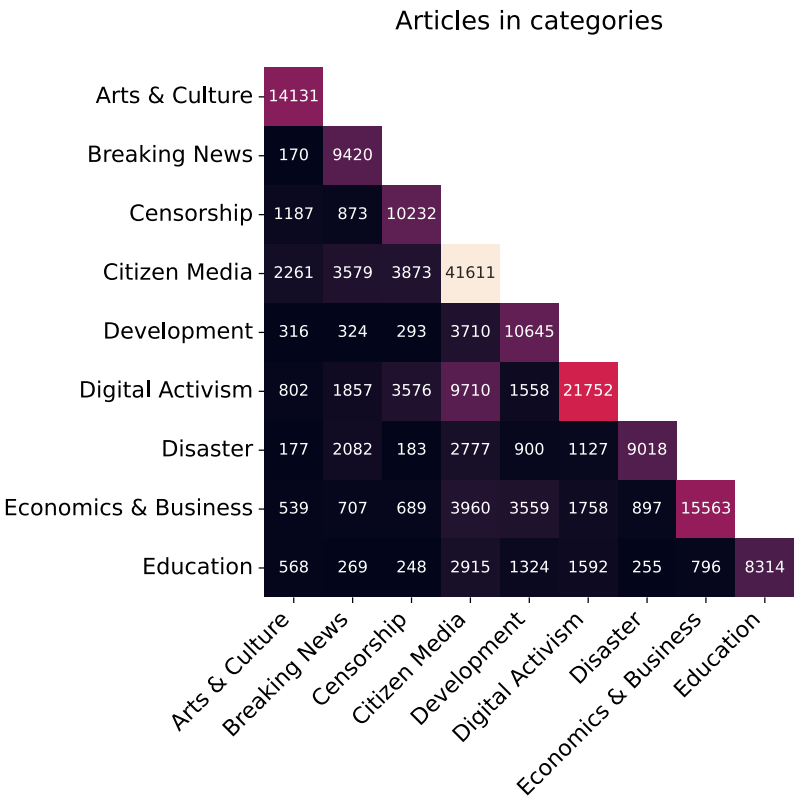

Figure 2.3: The correlation matrix shows how much the categories are correlated (how many articles are present in both categories). This is just a submatrix showing only 9 categories, where correlation can be seen, for example, between *Citizen Media* and *Digital Activism*.

This way, the final 25 categories were created. All of them should be distinguishable, and they should have a reasonable number of articles. There is one huge category, but even the smallest ones have more than 5,000 articles, as can be seen in Figure 2.5.

These categories can also be seen as an analysis of the most frequently occurring topics on Global Voices webpage. Articles are mostly oriented toward human rights, freedom of speech, and related topics like gender, LGBTQ+, governance, etc.

Some articles were lost, specifically those that belonged just to one of the deleted categories, but it is not a great number. The total loss is quantified after omitting languages with too few articles in the next section.

The second meta file is a JSON file, which serves mostly for keeping references to translations of articles. As explained above, the English version of Global Voices was scraped, and for each article, all its translations were downloaded as well. It is necessary to store all the articles together, and this is why in the second meta file, there are objects representing English articles. Every English article has a property of type dictionary, with language codes as keys and the names of articles as values. Regions are also stored for each English article. This way, it is possible to take all the English articles and have a link to all their translations. For the purpose of scraping additional Polish and Dutch texts, there are some texts in these languages at the end of the meta file. It is not a problem. To store just all English articles was chosen in order to avoid duplicates, and Polish and Dutch texts are not duplicates for sure. The reasoning behind this is described in the following text.

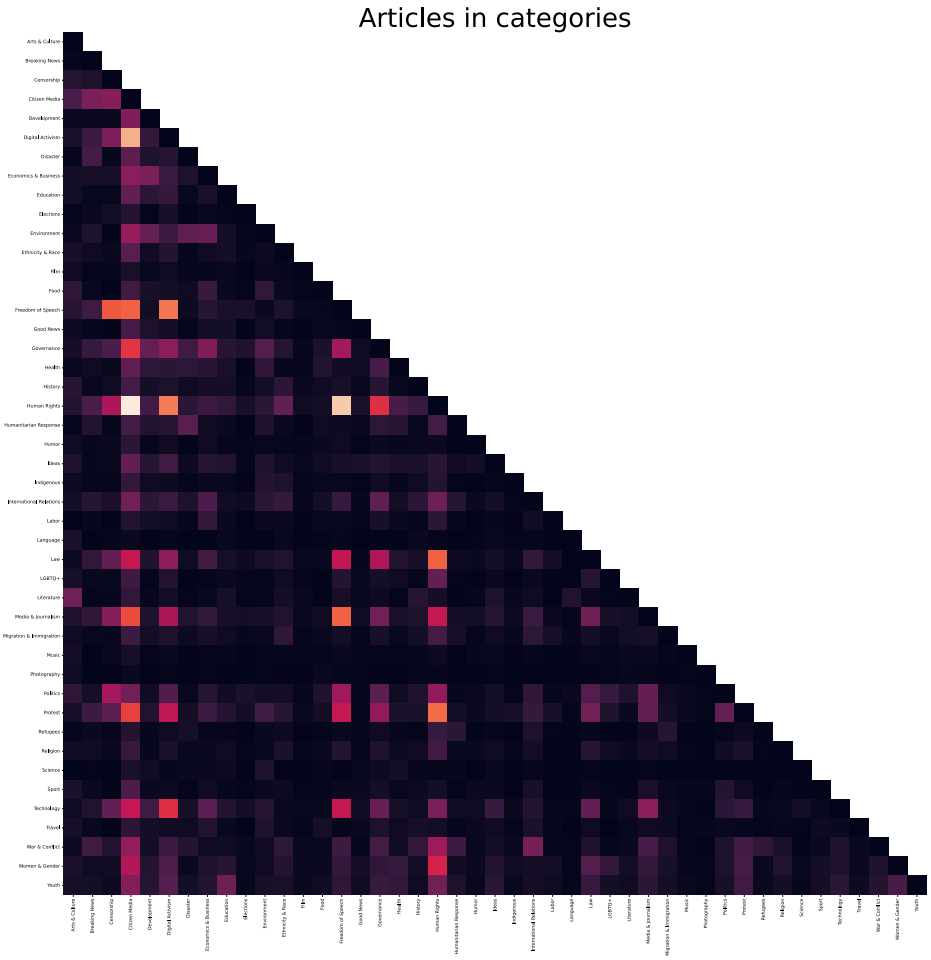

Figure 2.4: In this blind correlation map correlated categories are clearly visible. Diagonal was set to 0 in order to not affect the colors of heatmap. The lighter the color, the stronger the correlation. There are two rows that are highly correlated with almost every other category. The first one is in the fourth row; it is *Citizen Media*, and the second one is about in the middle; it is a category *Human Rights*.

Just as there were categories with a low number of articles, there were also languages with a low number of articles. Low numbers would not be sufficient for the desired tasks. In most of the cases, languages with a few texts were those for which searching for more articles would be almost impossible. Another problem would be combining the categories from another website with those from Global Voices. Most probably, it would be needed to map some categories or even merge some, which would decrease the number of categories. For this reason, low-article languages were simply omitted.

With respect to the Figure 2.6, languages with more than 2,000 texts were preserved. Since Dutch and Polish were very close to this limit, an experiment was performed. Although it seemed that all the articles were translated to English, after scraping specifically

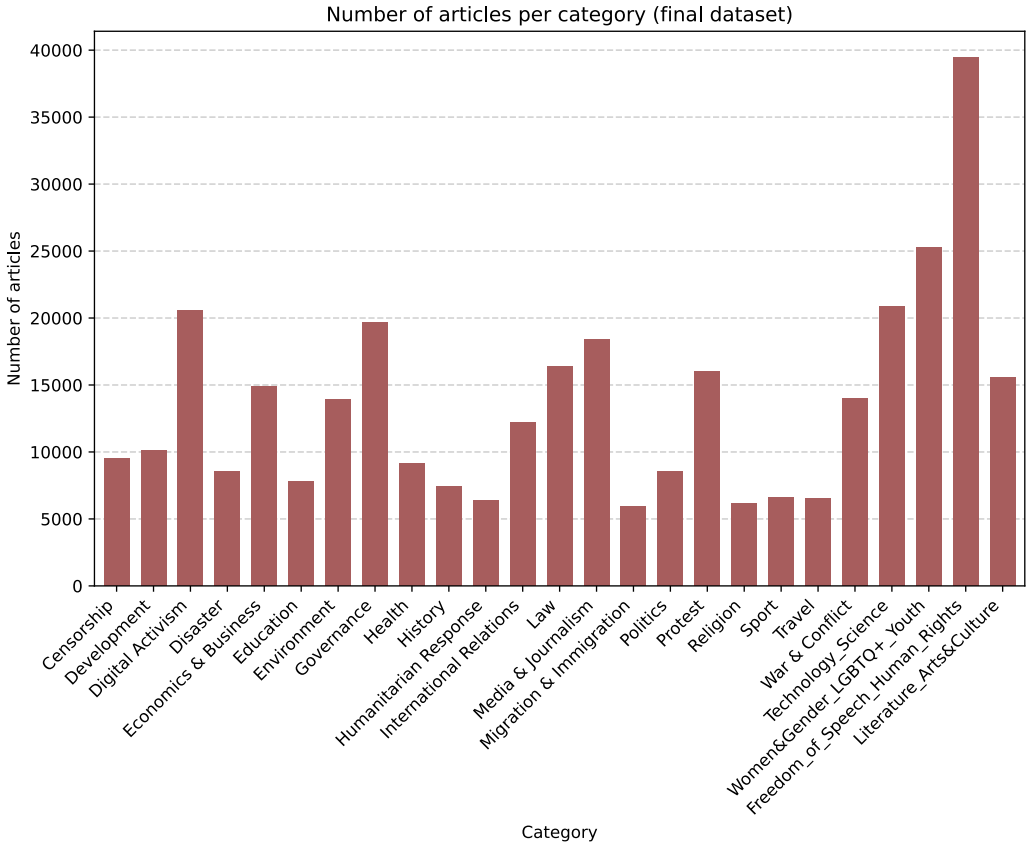

Figure 2.5: The final 25 categories after merging and deleting some of them.

the Dutch and Polish versions of Global Voices, some extra articles were discovered. The reasons behind this are two. Some of the articles weren't translated to English; the others were new, published during the period between scraping English version and the Dutch/Polish one. This way, also Polish and Dutch have moved into the category with more than 2,000 texts. Polish increased from 1,877 to 1,940 and Dutch from 1,863 to 1,915 articles. Languages with numbers of articles in the final dataset can be seen in Figure 2.7.

English significantly exceeds other languages in the number of articles, with almost 100,000 samples. This is caused by the fact that almost every article (with a few exceptions for Polish and Dutch) is translated into English. The matrix of translations can be seen in Figure 2.8. By deleting particular languages, some articles were lost. From the initial 245,821 downloaded articles, the number decreased to 232,318. It means 13,503 lost articles, which is about 5 %. This loss comprises articles lost due to category removal and also due to omitting languages.

Although for the purpose of this thesis some categories and languages were omitted, the original intention was to create a dataset useful for other tasks. For this reason, a complete chart with all the languages is available in the appendices.

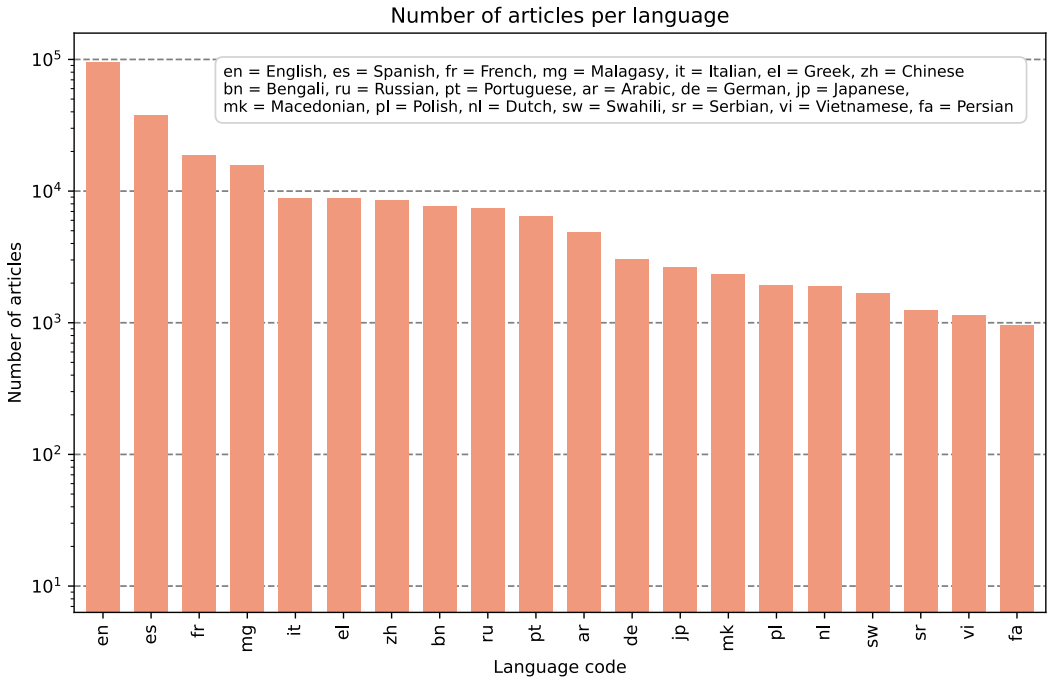

Figure 2.6: The plot shows the number of downloaded articles per language. Only twenty of the most numerous languages are displayed. English is the main language, with almost 100 000 articles out of 245 821 overall. The least numerous languages were Kurdish (ku) with 4 articles, Tetum (tet) with 8 articles, and Kazakh (kk) with 10 articles.

2.5 Dataset Structure

The whole dataset is structured as illustrated in Figure 2.9. Articles are stored language-wise. The names of the articles are taken from the URL address, specifically the last part of the URL address. For newer articles, the last part usually contains its name; for older ones, it is a number. It is not true for all languages; some language mutations have just numbers in the URL address. Regarding the structure of the Global Voices website, the number should be unique for individual languages; therefore, it is considered appropriate. Every article is saved as JSON and HTML with the same name. Meta files described above are stored separately.

2.6 Text Cleaning

The obtained dataset contains a lot of information for each article, like date of publication, date of downloading, title, description, language, URL, image URL, and others. For the purpose of this thesis, language, category, and main text are the most important. The main text is formatted as one long text sequence, where paragraphs are separated by a newline character. This format is convenient for further processing; there is just one complication.

The performance of multilingual models will be compared, so it is important that the text of the article is written in the expected language. Inconsistencies between the language and the language code given by the News-please library were corrected, as mentioned before.

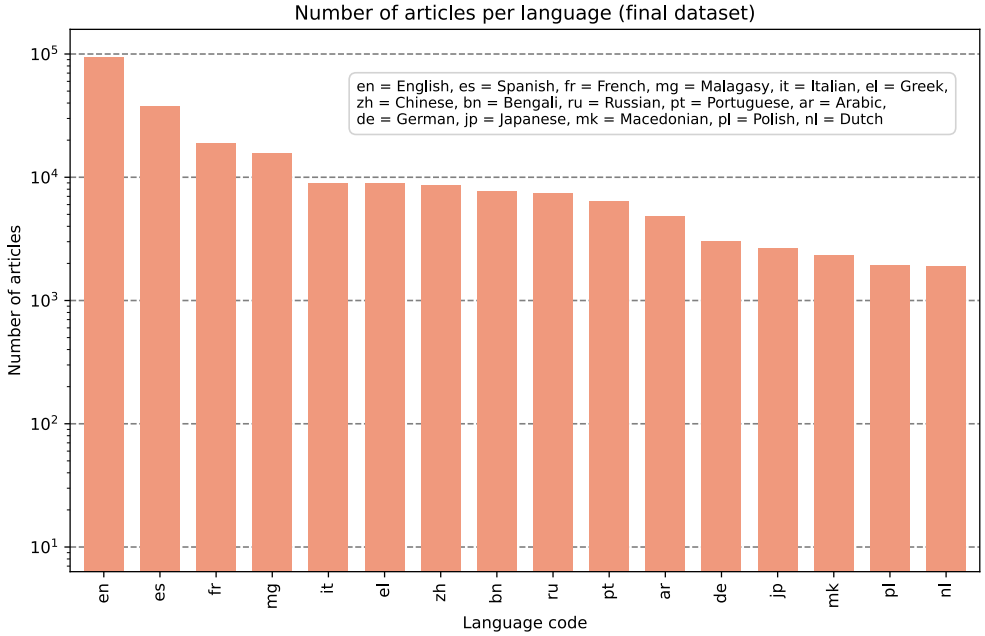

Figure 2.7: For the final dataset, 16 languages, which have more than 2,000 articles, were chosen.

Unfortunately, the language of the main text is not monolithic. The article is written in a particular language, but sometimes tweets in other languages (mainly English) occur in the main text. This would influence the results in cross-lingual experiments. If a language-specific encoder (for example, LASER3) was used to obtain embeddings, one English tweet would change the resulting embedding of the article; consequently, the tweets must be removed.

One possible solution would be to recognize the language and keep just the text written in the language of the article, although given the number and variety of languages, it might be challenging to find a library that is able to do it. Langdetect library [22] does not support Malagasy included in the dataset. Therefore, an easier and deterministic solution is introduced. In situations like this, .html versions of articles are useful. From the HTML source code, it is possible to extract additional information, including details about tweets.

One can take advantage of the fact that tweets are enclosed in HTML tags with the specific class *twitter-tweet*. The content of this tag can be taken and removed from the main text in .json file. As it is better to keep initially downloaded files in their original form, new .txt files containing just plain text without tweets were created.

2.7 Train/test split

While splitting data into train and test datasets, there is one rule that must be taken into account: the date of publication of the article. The year of publication is therefore the only criterion according to which the dataset can be split. This way, the same articles translated into different languages will be either in the train dataset or in the test dataset. Having the same article in both training and test data (just in a different language) is something

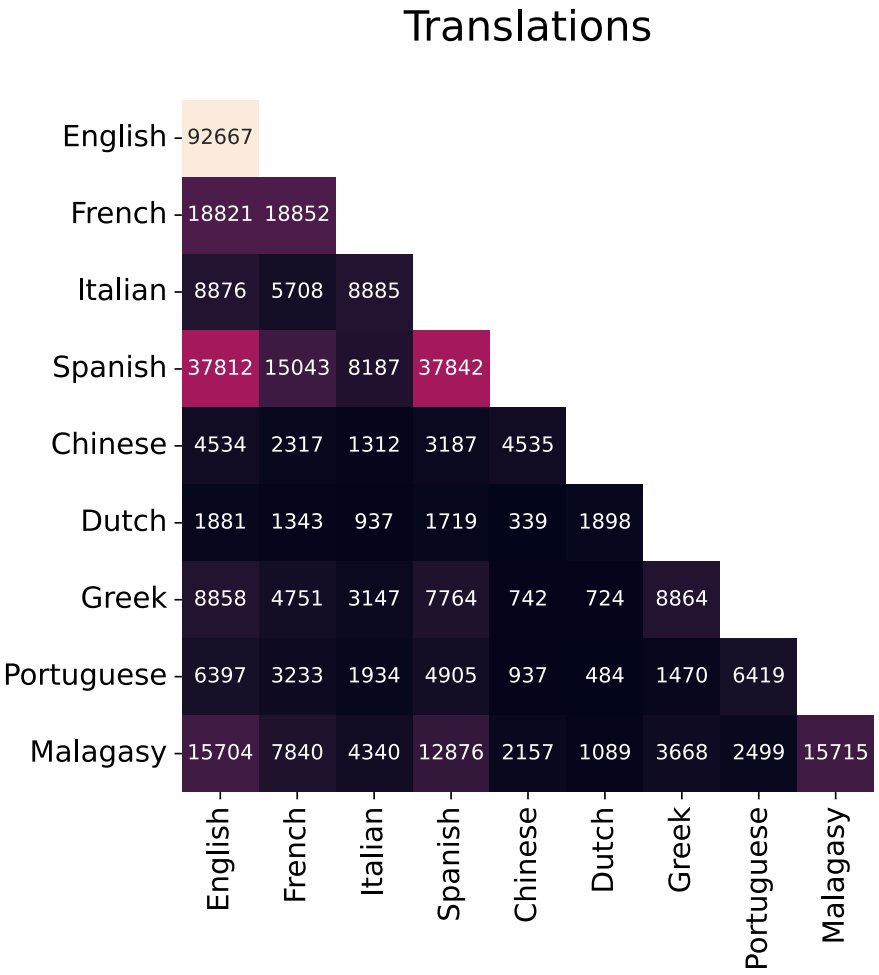

Figure 2.8: For most of the languages, almost all texts are translated into English. There are also a lot of texts translated into Spanish and/or French. On the diagonal, the total number of articles per language can be seen. This matrix shows just a subset of languages in the final dataset for better insight.

to avoid. Of course, there are marginal cases. Some articles were published at the end of the year and translated into other languages a few days later, in the following year. These cases were neglected because there were not many of them. There are over 8,000 articles in the entire dataset that were published and translated at the turn of the year, but only 300 texts in 2015/2016, the cut-off point for the distribution. This number was counted from the initial dataset with all the languages, including those that were not used in the end because of a lack of articles.

The main task, therefore, was to choose the cut-off point. In the dataset, there are articles from 2004 to 2022. As GlobalVoices was developing, in the beginning there were not all the currently present languages. It means that for some languages, there are no articles or a very low number of articles in the initial years. The distribution of articles through the years is different for every language; for some, the most articles were published in the last few years; for others, it is the opposite. An example of three languages is displayed in Figure 2.10. This affects the proportion of train and test articles. For this reason, it was not possible to follow the generally recommended ratio of train:test split

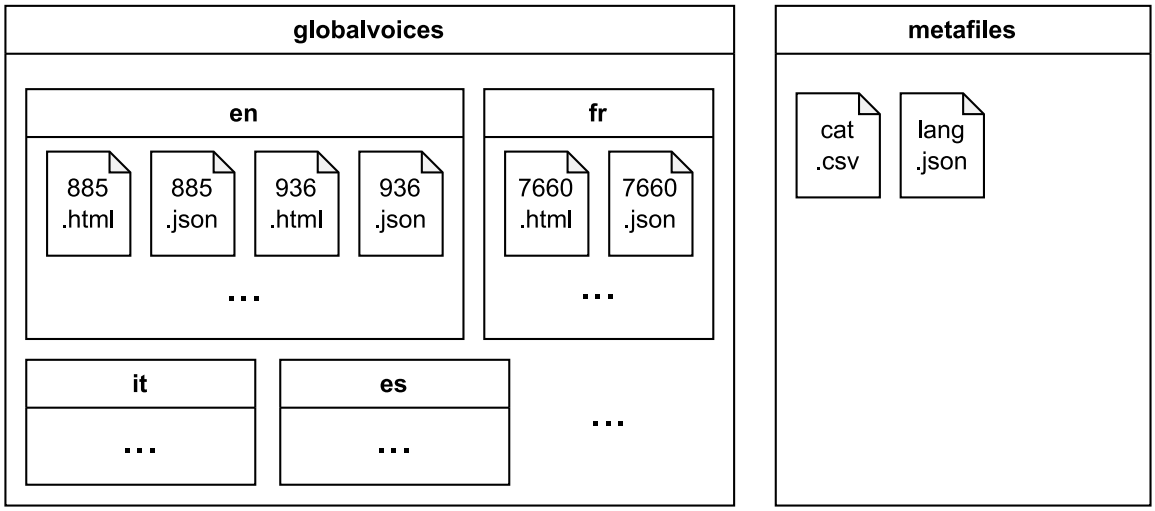

Figure 2.9: Folder *globalvoices* contains a folder for every language. In the language folder, the articles are stored in both HTML and JSON formats. Both formats have the same name, which corresponds to the last part of the URL address of the given article. Two metafiles, one with categories and one with translations, are stored separately.

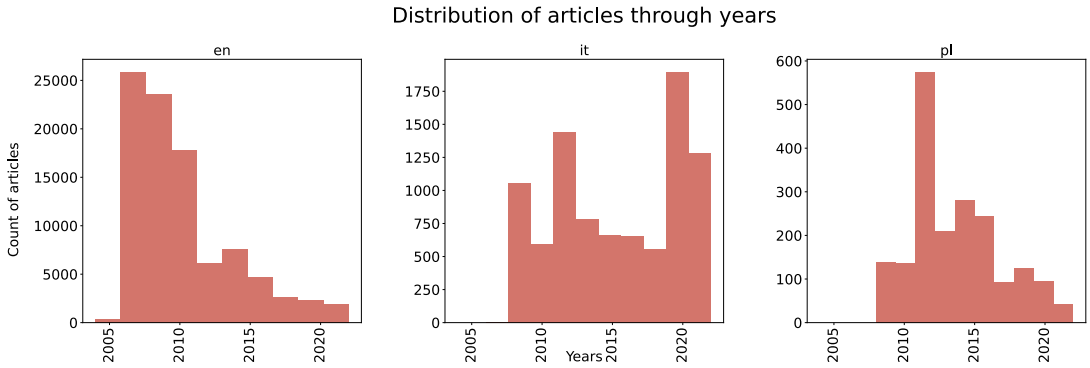

Figure 2.10: Example of three languages: Since the distribution of articles through the years differs for different languages, it is impossible to choose a perfect cut-off point.

80:20 or similar values. Sometimes it is rather 50:50, as illustrated in Figure 2.11. It was simply more important that all the languages are split in the same year.

The initial idea was to split the dataset into training, test and validation parts. It was very complicated to find two boundaries such that there are a reasonable number of articles for all the languages in the train, test and validation subsets. Moreover, while considering classifiers from the Scikit library, it turns out that some of them have the validation_split parameter directly, so it is not necessarily needed to split the dataset into three parts. For this reason, only the train:test split was performed in the end.

The initial estimation was that training data could be from 2004 to 2016, whereas test data could be from 2017 to 2022. This led to insufficient test data for some languages, so the final decision is training data until 2015 (including) and test data from 2016. The final train/test data ratio can be seen in Figure 2.11.

Languages probably have not changed that much over the years, but it is highly probable that some topics exist just in test data and not in training data. A good example is COVID-

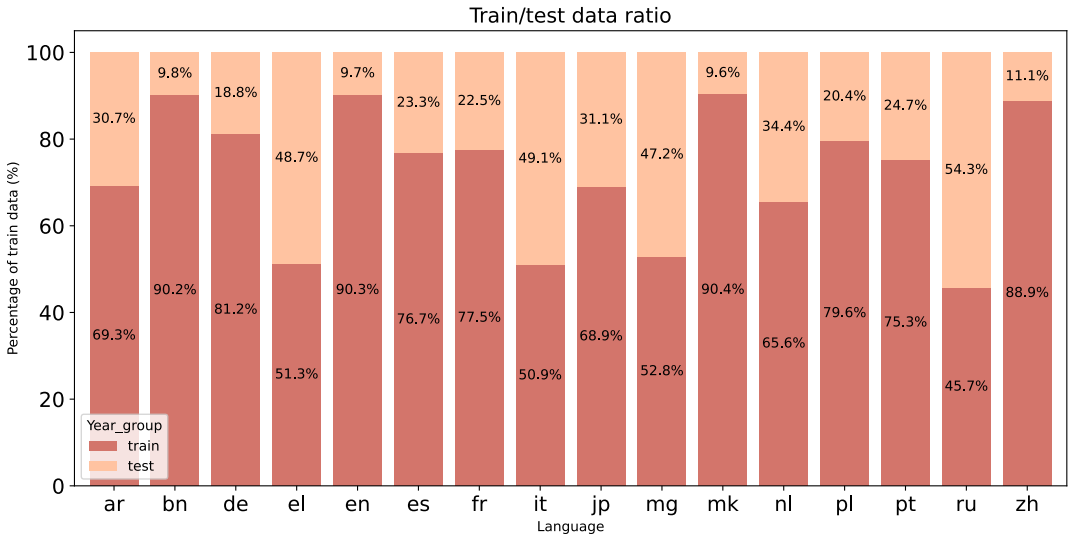

Figure 2.11: The ratio between tran and test set is not ideal for some languages, and with the given data, it was impossible to find it.

19, which is certainly not present in the training dataset. This particular problem was taken into account while gathering the dataset and its categories. GlobalVoices has a specific category named COVID; it is not used for this work precisely because there would be no training data for it, but the articles about COVID-19 would fall into other categories that were considered.

Chapter 3

Language Embedding Models

Natural language processing (NLP) techniques have made impressive progress in recent years. These methods require a huge amount of data, which limits the situations in which they can be used. For this reason, an approach consisting of learning general language representations on unlabeled data is popular. The first great success with this method is credited to word embeddings in Word2vec [12], followed by others, e.g., BERT [4]. This work was, however, surmounted by representations on the level of sentences. An early example is Sentence-BERT [20]. A model like this is able to encode sentences into a format that is then integrated into downstream systems specific to a given task.

Nevertheless, these approaches have one noticeable disadvantage. Models are language-specific; it is necessary to train a model for every language separately. This means not only a lot of tedious work but also low performance for low-resource languages [2]. For this reason, language-agnostic sentence embeddings were presented. These vector representations assure that sentences in different languages, which are semantically similar, are close to each other in embedding space. It brings the possibility of dealing with low-resource languages and cross-lingual transfer appplications; the model works well with them, even though it has not seen many samples during training. According to the authors of some language-agnostic models [8], the model is even able to work with languages that did not occur during the training phase but belong to a language family that is known to the model.

This thesis is comparing two popular language-agnostic models, LaBSE [5] and LASER [2], which are, at the time of writing this report, available in a version referred to as LASER3.

3.1 Transformers

Transformers are a type of deep learning model used in natural language processing (NLP) tasks such as language translation, sentiment analysis, and question-answering. They are made to overcome the drawbacks of earlier models like recurrent neural networks (RNNs) and convolutional neural networks (CNNs), which have trouble capturing long-range dependencies due to the vanishing gradient problem. Transformers recognize the contextual relationships between words in a sentence via self-attention techniques [23]. With the help of self-attention, the model is able to focus on various elements of the input sequence and determine the relative weights of the words in the sentence. As a result, transformers are able to represent long-range dependencies and the semantic connections among words in a sentence.

The input text is first tokenized into a sequence of individual tokens (e.g., words or subwords). These tokens are then passed through an embedding layer, which maps each token to a high-dimensional vector representation. Next comes the self-attention layer, which allows the model to access different parts of the input sequence and learn the importance of each token in the context of the sentence. Self-attention is achieved by computing attention weights between every pair of tokens in the input sequence. These weights are then used to compute a weighted sum of the token embeddings. The output of the self-attention layer is passed through a feedforward neural network that applies a non-linear transformation. This is followed by another self-attention layer, which further improves. The final output layer just maps vector representation of the input sentence to the desired output format (e.g., words, classification label). The most famous transformer-based language model is Bidirectional Encoder Representations from Transformers (BERT), which was introduced by Devlin et al., 2018 [4] and has achieved state-of-the-art performance on a wide range of NLP tasks. Since then, many other transformer-based models have been developed, such as GPT-3, RoBERTa or LaBSE.

3.2 LaBSE - Language-Agnostic BERT Sentence Embedding

LaBSE is a multilingual BERT embedding model that can produce sentence embeddings for 109 languages. The limit of input sentence is 512 tokens, and the resulting embedding dimension is 768. This is crucial for embedding extraction described in Chapter 4.

LaBSE uses a dual encoder architecture, so source and target texts (that are translations) are encoded individually using a shared transformer embedding network, which is initialized with an initial multilingual BERT checkpoint. The translation ranking task is then used to force similar representations for texts that paraphrase one another, while additive margin softmax helps to separate translations and nearby non-translations.

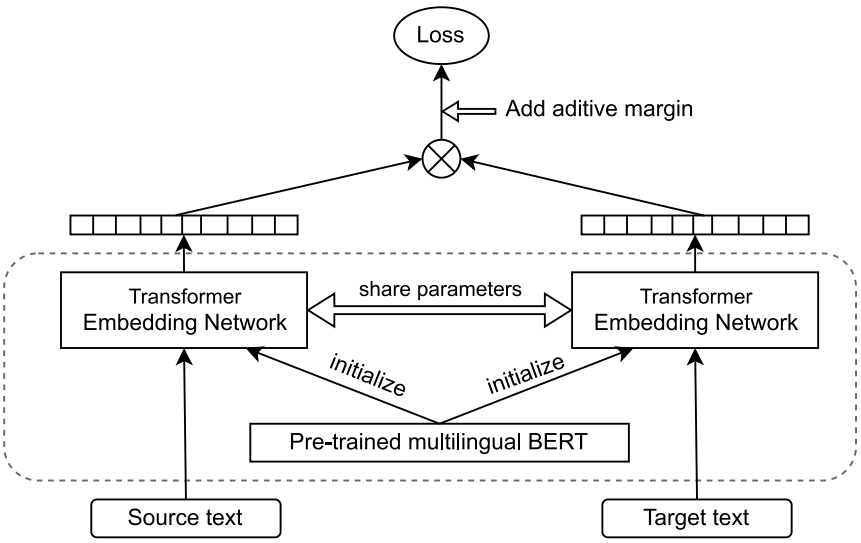

Figure 3.1: Labse uses pre-trained BERT to initialize dual encoder architecture, which share parameters. They encode source text and target text individually, then rank the translation with the help of additive margins for better separation of translations and nearby non-translations.

LaBSE is trained on monolingual sentences and bilingual sentence pairs while using masked language modeling (MLM) [4] and translation language modeling (TLM) [9]. In MLM, the model has a [MASK] token surrounded by context words and tries to predict what the [MASK] word is. With the TLM approach added, concatenated translation pairs are included.

The authors evaluated LaBSE mainly on bitext retrieval tasks [5]. They also performed a classification task on the SentEval benchmark, which, however, is in English only. The results showed that LaBSE, despite its massive language coverage, achieves competitive performance in comparison with monolingual embedding models. The authors have not tested performance in cross-lingual topic classification, and this is one of the goals of this thesis.

All the languages from the created GlobalVoices dataset are supported by LaBSE [5]. All of them were present in the training set of LaBSE, with English being the most numerous language (2 billion monolingual sentences). From the Figure 3.2 showing the quantity of training data in various languages, it can be seen that most languages of the Global Voices dataset are at the head of the plot, so there were many training samples. On the other hand, mg (malagasy) is rather in the tail, mk (macedonian), and bn (bengali) in the body of the plot; therefore, there were not so many training data samples. Despite this fact, the results of classification for these languages are not worse than for the most numerous languages.

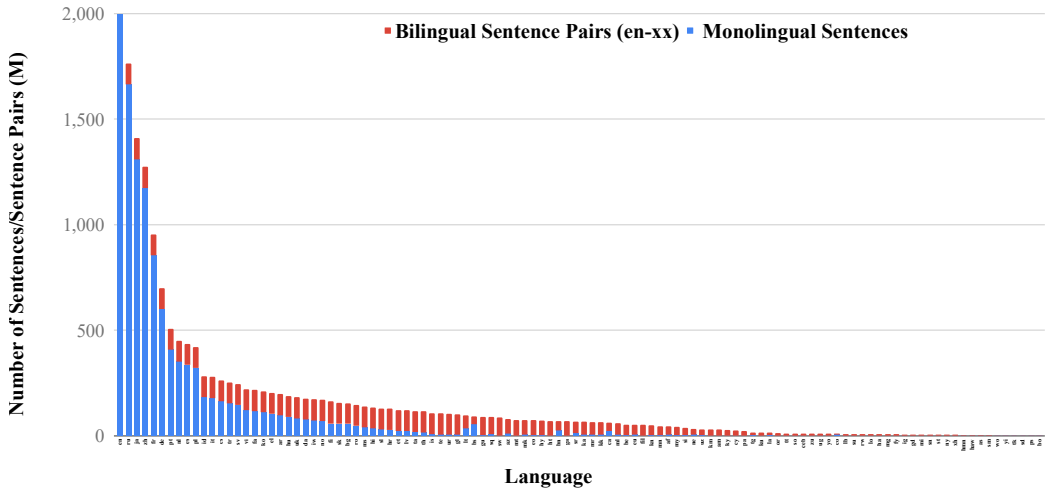

Figure 3.2: This plot shows quantity of monolingual sentences and bilingual sentence-pairs in the LaBSE training set [5]. According to the authors, there are 2 billion English sentences.

3.3 LASER (Language-Agnostic SEntence Representations)

On the contrary, the original LASER is not based on transformers. It was introduced to be able to solve tasks for languages with limited resources and to allow zero-shot transfer of a model from one language to another. It uses a sequence-to-sequence encoder-decoder architecture, where the BiLSTM encoder encodes the source sequence (with the use of byte-pair encoding (BPE) vocabulary) to a vector representation of fixed length, and the decoder creates a target sequence from this vector representation. In the end, the decoder

is discarded, and the encoder can be used to encode sentences. There is only one encoder, common for all languages [2].

Training was performed on corpora compiled from several sources. It minimizes the cross-entropy loss while iterating over all combinations of the involved languages. The authors decided not to use this approach when a sentence is translated into all other languages but to choose only two target languages. One of the reasons was the quadratic cost of combining all the languages; the second was the missing N-way parallel corpus. It is precisely this problem of searching multilingual datasets that LASER aims to solve. LASER can generalize to languages that were not encountered during training but are members of a family that includes covered languages.

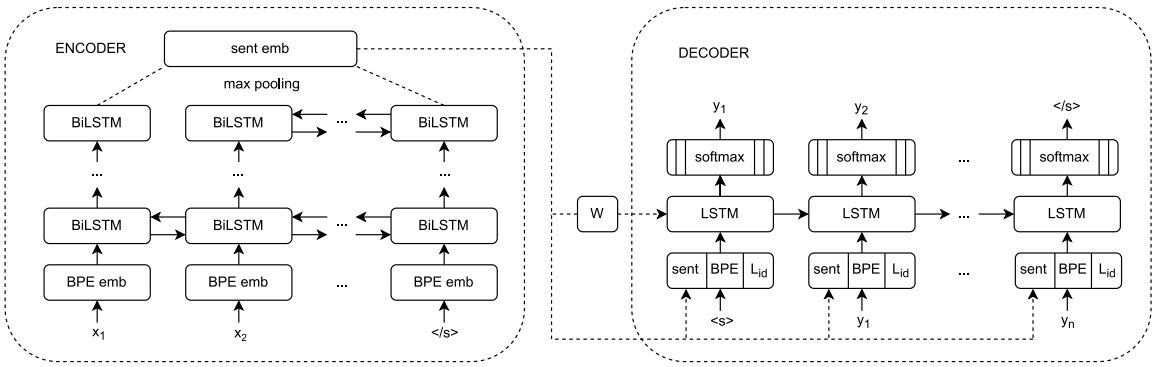

Figure 3.3: LASER uses an encoder-decoder architecture. Sentence embeddings are yielded by the final representation or a pooling mechanism. The decoder LSTM is initialized with these embeddings through a linear transformation.

The LASER model encodes sentences in a text as a list of vectors of length 1024. The hidden dimension is 512, and concatenating the hidden dimension representations from both directions gives 1024. Unlike LaBSE, LASER can process arbitrarily long inputs. It supports 93 languages. LASER2 changed the tokenization from BPE to unigram based subwords, and for the project No Language Left Behind [15], the authors developed new encoders, referred to as LASER3. LASER3 is based on transformers architecture, and trained using knowledge distillation. LASER3 works with more than 200 languages. It makes use of a common space to incorporate sentences in any language. According to its authors, it supports code-switching (words from multiple languages can appear in the same sentence). For LASER3, however, it is important to specify the input language. If the language is not specified, an older version of LASER is used. Since languages added in version 3 are not covered in the obtained dataset, this work is technically not using LASER3 but LASER2. In any case, LASER3 does not mean anything new in the architecture, just newly supported languages (mainly African and other less common languages).

To summarize, LaBSE is a transformed-based model trained on a huge amount of data. LASER is not transformed-based; it consists of an encoder and a decoder, and it was pre-trained on parallel data from multiple languages.

Chapter 4

Embedding Extraction from Documents

After finalizing the dataset, the main text of the articles still has to be transformed into embeddings, into numerical representation. For this, two different models discussed in Chapter 3 (LaBSE and LASER) are used. The extracted features are afterwards used as input to a logistic regression and other classifiers described in Chapter 5.

The embedding extraction is done for every article using both the models, embeddings are stored in separate files and used for training with different classifiers and then also for topic discovery task.

Articles from the created dataset can have several long paragraphs, whereas LaBSE can handle inputs of length at most 512 tokens. There are several possibilities for dealing with this length problem. For the classification task, the usual solution is to divide long text into several smaller paragraphs, and then every paragraph is transformed into embeddings. Finally, an average of these embeddings is taken for each document.

4.1 LaBSE and LASER embeddings

Pre-trained transformers such as BERT or LaBSE are used in a wide range of NLP tasks, including text classification. The contextual word embeddings generated by these pre-trained transformers are advantageous when handling words with multiple meanings and other NLP challenges. Similar things can be said about LASER.

The goal of this thesis is to benchmark the performance of these models on the newly created dataset. Therefore, embeddings for articles are created using both of them. Both variants are then used for classification and the following tasks.

At first, LaBSE embedding for each article was stored in a separate file. Loading tens of thousands of articles while training was computationally demanding; therefore, for the following work, a better strategy is proposed. All embeddings should be stored in two files: one for training embeddings and one for testing embeddings for every language.

LaBSE embeddings can be obtained straightforwardly thanks to the Python framework SentenceTransformers. This way, it is fairly effortless to load information directly from the json file or from the cleaned txt file mentioned in Chapter 2. The resulting embeddings have 768 dimensions.

The LASER model is not ported into any easy-to-use Python package, yet it is available on GitHub, and the authors prepared a script for encoding texts into embeddings. The

script expects text in the input file, which is then tokenized. Language can be specified while running the script, but language-specific encoders are available just for LASER3. If the language is not specified, the script defaults to LASER2. Since the gathered dataset does not contain any of the languages added in LASER3, the original LASER2 encoder supporting 93 languages is used. For this model, the resulting embedding has 1024 dimensions.

One of the primary challenges in text classification using the above-mentioned models is the length of the documents. Due to the maximum input size limitation of 512 tokens for BERT transformers, it is unlikely that the entire document can be processed at once. To address this issue, various approaches are available.

In some fields, it has been discovered that achieving high classification accuracy does not necessarily require using the entire text. The solution is to choose only a specific portion of the text (e.g., the beginning or end, or both, or the title) that can fit into the model's input layer. Taking into account the available data and their nature, it is possible to use the title and text of the article. Three different approaches were adopted.

4.2 Titles

The first strategy consists of embeddings extraction based on the title of the article. It is a short text; on the other hand, it should briefly but succinctly describe the main topic of the article. The title is always present; each article has its own title. That is why there are some inconsistencies in the number of training and testing examples between titles, averages, and truncations of the main text. Some old articles obtained from GlobalVoices have just the title and no content. The difference between the number of articles is not very significant; usually it is just dozens of articles. Another advantage of using titles is that models are able to handle the whole title. The title is never longer than 512 tokens, unlike the body of text, which is the problem addressed in the next part.

4.3 Truncation of main text

The second strategy uses the main text of the article. Models are not able to handle a whole long text; the maximum input size for LaBSE models is 512 tokens; therefore, text passed to a model is truncated. 512 tokens may not be enough to express the topic of the article; sometimes the perex (first paragraph) is just an introduction and does not carry a lot of useful information.

4.4 Average of paragraphs

Taking information from the whole text of the article appears to be more promising. One of the proposed strategies consists of computing the average embedding for the whole text of the article. Every paragraph is treated individually, and embedding for this paragraph is created. This means that if there are some very long paragraphs, they are truncated, similarly to what happens in the above-mentioned strategy of using the beginning of the main text. In the end, the average value of all these embeddings is stored.

This approach may seem like the most promising technique. It can include all the topics mentioned in the text, which is not the case when taking just the beginning of the article. For example, in the article about women who launched a fashion business from prison, you

can speak about prisons in general and women in prisons in the first paragraph but then talk more about business strategies. The first paragraph means that this article belongs to the category *Women & Gender*, the second one falls rather under *Economics & Business*. If only the beginning of the article is used, the information about business is lost.

On the other hand, there are still some drawbacks, mainly that there may be many subtopics mentioned in different paragraphs, and these butopics may not even belong to any of the classes (categories). Averaging the embeddings might result in a loss of fine-grained semantics. However, we believe such information would not be significant for coarse-level topic identification.

Chapter 5

Cross-lingual category classification

Classification is a supervised learning task to identify the category of new observations. Along with the clustering described in Chapter 6, these are examples of the general problem of pattern recognition. How well this problem can be solved depends both on the quality of sorting articles into categories and the quality of their representation (embeddings). One of the aims of this thesis is to compare LaBSE and LASER models, whereby classification gives some insight into how the models perform and how well they are able to encode texts.

This chapter describes the classification task performed on the obtained dataset. At first, it discusses difficulties, which are determined by the nature of the data, and suggests techniques that must be used due to this. Then it considers different downstream classifiers typically used for such tasks. A great part of this chapter is dedicated to presenting the results and comparing them not just between various classifiers but also between embedding models. Both models are performing well on all languages presented in the dataset.

5.1 Multiclass and multilabel classification

The dataset of articles from the GlobalVoices website brings two main challenges that must be taken into account:

1. Multi-class classification: There are not only two classes; there are many categories, in particular 25 categories. Although this may seem like a big number, it has to be mentioned that the initial number of categories was even higher. Some categories were merged, as described in Chapter 2. The decisions about what classes to merge were made in such a way that there are no different categories that are similar since the similar categories were merged. In any case, there might still be some relationships between categories that make it difficult to distinguish between them. After all, these relations are expressed by classifying the article into multiple categories, which is complication number two.

2. Multi-label classification: Single-label classification is the case when every sample belongs to just one class. This is not the case with Global Voices articles. It may happen that an article is only assigned to one category, but in general, this does not hold true. Data samples are not mutually exclusive.

Characteristics of these types of problems are displayed in a Venn diagram in Figure 2.1. In most cases, multi-labels are also multi-class classifiers.

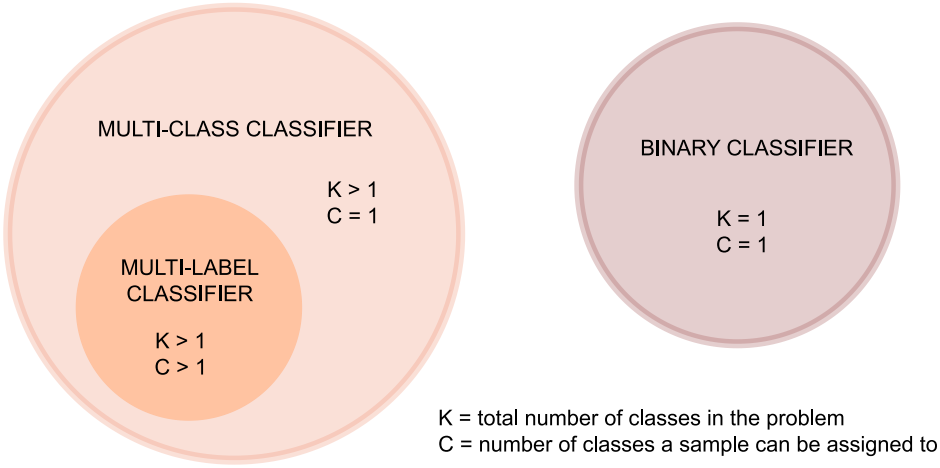

Figure 5.1: The problem of article category classification is multi-class and multi-label. It means that there are more than two categories (classes) in the problem and that every article can be assigned to more than one category (class).

Not all the classification algorithms natively support multi-class cases, yet there are several solutions to adapt the problem. One approach is to split multi-class classification into several binary classifications and fit a binary classification model on each of them [19]. This can be implemented as One-vs-Rest or One-vs-One.

- One-vs-Rest strategy (sometimes calles as One-vs-All) consists of splitting the dataset into multiple binary classification problems. In the case of the GlobalVoices dataset used for this thesis, it means training 25 separate classifiers (one for each category). Each classifier is able to decide if an article belongs to the category or not. In other words, C binary classifiers are trained, where the data from class C is treated as positive and all the other data is treated as negative [14] Because of that, the dataset is unbalanced.

- One-vs-One also splits the dataset into binary classification problems; this time it creates one binary classifier for each pair of classes. It results in creating K*(K-1)/2 classifiers, where K is the number of classes. For the GlobalVoices dataset, it would be 25*24/2 = 300 classifiers. This is also one of the reasons why One-vs-Rest approach is used in the experiments described below.

This way, it is possible to use algorithms designed for binary classification like Logistic Regression, Multilayer Perceptron or Support Vector Machines. The examples of multi-label classifiers that do not require problem transformation are neural networks with Backpropagation for Multilabel Learning[25] or decision trees.

5.2 Classifiers

Due to the above-mentioned One-vs-Rest strategy, it is possible to use a simple binary downstream classifier. This work is focused mainly on Multilayer Perceptron (hereafter referred to as MLP), Support Vector Machines (SVM) and some experiments were also performed with Logistic Regression (LR). All of them are implemented in the Scikit-learn library [18], the MLP version created with the PyTorch framework [17] was tested as well.

An artificial neural network (ANN) called a multilayer perceptron (MLP) is made up of numerous layers of interconnected nodes, or neurons. As a feedforward neural network, it transmits information from the input layer via one or more hidden layers and then to the output layer. The data is received by the input layer, calculations are done by the hidden layers, and the prediction made by the network is created by the output layer.

One disadvantage of MLPs is that they can be prone to overfitting, especially when the number of hidden layers and neurons is large. In this case, regularization techniques, such as weight decay and dropout, can be used to prevent overfitting. Since the MLP classifier implemented in Scikit-Learn does not support setting dropout, another version was implemented using the PyTorch framework.

Fortunately, it turns out that the inability to set a dropout does not degrade the results; the numbers are similar for both versions of MLP classifiers, as can be seen below.

The MLP classifier implemented in PyTorch makes it necessary to implement the One-vs-Rest approach manually. This means training 25 separate classifiers (one for each category) fed with modified data. Instead of taking existing labels, it was needed to relabel data in such a way that a vector of labels becomes one single value, 1 or 0, depending on whether the article belongs to the currently trained category or not. Training 25 classifiers for each of the 16 languages means a total of 25*16 = 400 classifiers. That is challenging to manage; therefore, using Scikit implementations with the existing OVR approach is a lot more convenient.

SVM is another classifier used for category classification [6]. SVM works by constructing a hyperplane that separates the data into different classes. The hyperplane is chosen so that it maximizes the margin between the two classes, which is the distance between the hyperplane and the closest points from each class. SVM uses a kernel function to map the input data into a higher-dimensional feature space, where it is more likely to be linearly separable. This allows the SVM to find a hyperplane that separates the data even when it is not linearly separable in the original input space. Because it can successfully handle high-dimensional data, it is particularly helpful in jobs where the number of features is high relative to the number of samples. Moreover, SVMs are not very sensitive to the existence of outliers in the data. SVMs have the drawback of being sensitive to the regularization parameter and kernel function selection, which can degrade method performance. Nevertheless, this was not observed.

Logistic regression is a statistical model that is commonly used for binary classification tasks, which is the case with the OVR approach in this work. The output of this model is a logistic function of a linear combination of the input features. The logistic function, also referred to as the sigmoid function, maps any input to a value in a range from 0 to 1, which represents the probability of the binary outcome. The logistic regression model is trained using a maximum likelihood estimation approach, which involves optimizing the model parameters to maximize the likelihood of the observed data.

As both MLP and SVM were giving fairly good and comparable results, linear regression was only used as a test. Results were significantly worse, so testing was not pursued in this direction, leaving the previous two classifiers.

5.3 Metrics

In order to assess how well a categorization model is performing, there are several commonly used metrics. These metrics offer several viewpoints on a classification model's effectiveness and can be used to evaluate different aspects of the model's behavior. It's crucial to select the right metric(s) based on the requirements for a specific problem. Some of the most popular measures are listed below:

- Accuracy: Accuracy measures how many of the model's predictions were correct. The number of correct predictions divided by the total number of predictions is the simplest and most intuitive classification metric.
- Precision: Precision measures the proportion of true positives (correctly assigned positive predictions) out of the total predicted positives. It is defined as the number of true positives divided by the sum of true positives and false positives. Recall: Recall measures the proportion of true positives out of the total actual positives. It is defined as the number of true positives divided by the sum of true positives and false negatives.
- F1 score: The harmonic mean of precision and recall, which balances both measures, is the F1 score. It is specified as 2*(precision*recall)/(precision+recall).
- Area under the ROC Curve (AUC): The ROC (receiver operating characteristic) curve shows the true positive rate (recall) against the false positive rate (1-specificity). AUC provides a single scalar value to assess the overall performance of the model and measures the area under the ROC curve.

Accuracy is the simplest method, but it ignores class imbalance. It is misleading for imbalanced classes, and since the OVR approach is used, labels are positive for one category and negative for 24 other categories. This means the dataset is highly imbalanced. Also, it does not reflect the cost of different types of errors, even though article classification is not the case where false negatives are more harmful than false positives, as it is, for example, in medical diagnosis. In any case, accuracy was rejected. Among the other metrics, the F1 score was chosen as it reflects both precision and recall.

Considering the class imbalance, weighted F1 is used, as it takes a weighted average based on the number of samples for each class. More weight is given to classes with more samples; each class contributes to the final score proportionally. Someone could suggest that for cases with classes equally important, the macro F1 score is a better choice. This can be discussed. Some categories are occurring more often, and they will be occurring more often even in the real data when the model is in production, so it may be good to give importance to class imbalance.

5.4 Zero-shot learning

Zero-shot learning (for the first time introduced as dataless classification [3]), in general, is a principle when a model is used to perform something it was not explicitly trained for.

In other words, it allows a machine learning model to generalize its understanding of the world to new and unseen objects without the need for any additional training data. An example can be a classifier trained on images of animals, which is able to recognize unseen animals thanks to learned knowledge, or a machine translation system, which can also be used to translate to and from languages that it has never seen before.

Zero-shot learning has the potential to enable machine learning models to recognize and understand the world in a more flexible and generalizable way, without the need for large amounts of training data. In the context of natural language processing, it also means that there is no need for training data for some low-resource languages. A model can be trained, for example, on English texts, which are easily available and used for other languages in production.

Zero-shot learning was used in the below-mentioned experiments. A classifier was trained on only one language and tested on all the languages present in the dataset. In this case, zero-shot learning is performing very well. The model is able to generalize, and the classification works great even for languages that are not present in the dataset at all. This is not part of the experiments, but it can be seen in the web application. It classifies accurately even articles in the Czech language and others, on which the LaBSE model was trained.

5.5 Experiments and results

During the experimenting phase, all the above-mentioned classifiers were trained along with various types of embeddings (titles, truncation of maintext, and average of paragraphs). For each combination of classifier and embedding type, 16 models were trained (one model per language). These models were tested on all languages. This means 16*16 = 256 tests. Due to the huge number of results, only selected results are presented below. Every experiment is evaluated separately for diagonals and non-diagonals. The numbers on the diagonal mean that the model was trained and tested on the same language; therefore, it does not make sense to mix them with numbers representing training and testing on different languages. All the below-reported numbers are weighted F1 scores, as explained above, with the exception of Table 5.3, which talks about macro F1 scores.

Table 5.1 shows the overall results for all experiments. Weighted F1 is computed separately for every pair of languages; therefore, there are 256 numbers. Then an average over diagonal and non-diagonal is taken. The diagonal represents cases when the model was trained and tested on the same language, so it is interesting to separate it from the rest. Anyway, the numbers for diagonal and non-diagonal (zero-shot) are very similar, which is good news because it proves that multilingual embeddings are well created.

All the experiments were performed initially on LaBSE embeddings. At first, it was necessary to choose the best classifier. MLP with two layers of 768 neurons (the size of LaBSE embedding) gives the best number. Due to the inability to set the dropout parameter for the Sklearn implementation of MLP, MLP written in PyTorch was also tested; it gave more or less similar results, not better ones. SVM is performing slightly worse, but the differences are vastly small. The only classifier giving significantly worse results was LR, which is why it is not included in this table. Weighted F1 was about 0.6, so this experiment was not pursued.

Since MLP gave the best results for LASER embeddings, MLP with two layers of 1024 neurons (the size of LASER embeddings) was tested at first. Results are almost the same as

Weighted F1		Title Embedding		Truncation Embedding		Average of Paragraphs Embedding	
		Diag.	Zero-shot.	Diag.	Zero-shot.	Diag.	Zero-shot.
LaBSE OVR MLP (512, 256)	Avg	0.8436	0.8406	0.8264	0.8253	0.8569	0.8522
	Std Dev	0.0101	0.0131	0.0286	0.0217	0.0158	0.0169
LaBSE MLP (768, 768)	Avg	**0.8669**	**0.8644**	0.8446	0.8420	0.8753	0.8728
	Std Dev	**0.0068**	**0.0113**	0.0084	0.0115	0.0182	0.0191
LaBSE SVM	Avg	0.8204	0.8134	0.8328	0.8342	0.8559	0.8475
	Std Dev	0.0133	0.0142	0.0356	0.0287	0.0090	0.0152
LASER MLP (1024, 1024)	Avg	**0.8677**	**0.8651**				
	Std Dev	**0.0071**	**0.0113**				

Table 5.1: The table shows the weighted F1 score and standard deviation for different experiments with various classifiers and both embedding models. The numbers prove that not only the classifiers are giving very similar numbers, but also embeddings created with LaBSE and LASER models work comparably well. A green background represents the best numbers. Bold text represents the numbers to be compared.

for LaBSE embeddings; therefore, it can be concluded that both embeddings are performing equally.

Considering various strategies for creating embeddings, the average of paragraphs gives the best results, even though there is just a diminutive difference.

Table 5.2 shows detailed information for the MLP classifier with two layers and LaBSE embeddings created with the average of the paragraphs approach. It is just a subset of the whole table, which can be seen in the appendices. The most important thing is that the standard deviation is not high, so the F1 scores are quasi-equal for all the languages.

For comparison, there is a Table 5.3 showing macro F1 scores for the same experiment (LaBSE average embeddings, 2 layers MLP classifier). The numbers are significantly lower (the difference is approximately 10 %). This shows that the model is performing poorly on some of the classes, which have fewer samples. Macro F1, unlike weighted F1, does not put more emphasis on the classes with a larger number of samples. Thus, a poor performance in the smaller classes brings the macro F1 score down.

The Figure 2.5 displays the number of articles in various categories. From this plot, it could be predicted that, for example, International Relations, Migration & Immigration, and Religion categories could be the cause of poor macro F1 scores. For confirmation of this theory, F1 scores for individual categories are displayed in Table 5.4. It turned out that the number of articles in the category may not play a great role. For example, Politics category shows a worse F1 score than Health. There are $30,705+13,892+6,910+798+667+594 = 53,566$ articles in the Politics category (with respect to en, es, fr, mk, nl languages) and $3,970 + 1,904 + 1,119 + 157 + 136 + 133 = 7,419$ articles in the Health category.

Weighted F1		TESTED ON												
		en	**es**	**fr**	**mg**	**it**	**el**	**zht**	**zhs**	**...**	**mk**	**pl**	**nl**	**AVG**
TRAINED ON	**en**	0.884	0.883	0.881	0.893	0.882	0.885	0.895	0.908	...	0.898	0.882	0.885	0.886
	es	0.881	0.886	0.885	0.886	0.885	0.887	0.888	0.898	...	0.901	0.887	0.889	0.887
	fr	0.874	0.883	0.884	0.884	0.882	0.884	0.885	0.900	...	0.893	0.890	0.892	0.884
	mg	0.884	0.885	0.884	0.882	0.884	0.886	0.893	0.906	...	0.899	0.883	0.888	0.887
	it	0.874	0.881	0.881	0.879	0.880	0.882	0.882	0.890	...	0.892	0.877	0.884	0.880
	el	0.826	0.839	0.835	0.841	0.837	0.833	0.849	0.859	...	0.852	0.841	0.844	0.841
	zht	0.880	0.881	0.878	0.879	0.880	0.880	0.894	0.933	...	0.883	0.870	0.882	0.882
	zhs	0.873	0.874	0.872	0.875	0.873	0.873	0.898	0.893	...	0.883	0.869	0.879	0.875
	bn	0.865	0.873	0.872	0.868	0.870	0.873	0.878	0.886	...	0.880	0.876	0.879	0.874
	ru	0.868	0.886	0.889	0.872	0.884	0.887	0.881	0.887	...	0.907	0.893	0.894	0.886
	pt	0.869	0.880	0.880	0.879	0.878	0.880	0.881	0.888	...	0.885	0.877	0.886	0.879
	ar	0.883	0.886	0.887	0.878	0.886	0.886	0.896	0.912	...	0.887	0.880	0.892	0.886
	de	0.868	0.879	0.880	0.872	0.878	0.881	0.884	0.888	...	0.889	0.878	0.888	0.879
	jp	0.863	0.867	0.867	0.861	0.868	0.869	0.869	0.880	...	0.883	0.871	0.877	0.870
	mk	0.826	0.841	0.839	0.839	0.839	0.836	0.852	0.858	...	0.854	0.840	0.849	0.844
	pl	0.855	0.863	0.861	0.848	0.861	0.857	0.865	0.869	...	0.871	0.864	0.869	0.862
	nl	0.822	0.833	0.834	0.834	0.832	0.830	0.853	0.862	...	0.843	0.839	0.839	0.838

Table 5.2: This table displays the weighted F1 score for the best experiment (even though the differences are really small), LaBSE embeddings (average of paragraphs), in combination with the MLP classifier (two layers of 768 neurons). The standard deviation of numbers is very small (as illustrated in the previous table), and the worst classifier here is the one tested on NL articles. Low performance can be explained by the small number of articles in this language. In this experiment, the best classifiers are the ones with a greater number of articles (es, mg). The green background represents the best F1 for each column; therefore, for example, en was classified the best with a classifier trained on the en and mg datasets.

.

Table displays category-wise F1 score for 3 most numerous languages and 3 least numerous ones. These results belong to two layers MLP implemented with PyTorch, which did not seem to work better than MLP from Scikit-learn, but it performs better when focusing on macro F1. For some categories it is rather obvious why the F1 score is high. For example Sports or Health category can be distinguished from the others, on the other hand Politics may be easily missclassified. This should be consistent with results of clustering. The aforementioned Sport could be further from other categories.

One extra experiment was performed on a lower number of articles. Since for the least numerous languages, the dataset contains about 2000 articles, this count was chosen. Table 5.5 displays weighted F1 scores for the same classifier and parameters as Table 5.2. The only difference is that for training, only 2000 samples were used. Test samples were preserved, which means that for some languages there were a lot more test samples than train samples (e.g., for English, there were 8800 test samples and 2000 train samples). The

Macro F1		TESTED ON												
		en	**es**	**fr**	**mg**	**it**	**el**	**zht**	**zhs**	**...**	**mk**	**pl**	**nl**	**AVG**
TRAINED ON	**en**	0.748	0.747	0.743	0.756	0.744	0.743	0.734	0.761	...	0.741	0.732	0.732	0.739
	es	0.734	0.748	0.746	0.731	0.748	0.741	0.713	0.748	...	0.745	0.749	0.741	0.738
	fr	0.722	0.747	0.748	0.725	0.746	0.740	0.704	0.732	...	0.745	0.760	0.751	0.737
	mg	0.746	0.759	0.757	0.727	0.755	0.752	0.725	0.751	...	0.758	0.726	0.748	0.745
	it	0.714	0.741	0.744	0.705	0.740	0.733	0.693	0.707	...	0.730	0.724	0.734	0.724
	el	0.757	0.780	0.786	0.723	0.780	0.756	0.728	0.740	...	0.788	0.794	0.789	0.770
	zht	0.744	0.733	0.730	0.712	0.734	0.724	0.740	0.834	...	0.684	0.681	0.721	0.721
	zhs	0.723	0.722	0.718	0.704	0.718	0.711	0.744	0.722	...	0.707	0.689	0.714	0.710
	bn	0.690	0.714	0.715	0.686	0.708	0.707	0.688	0.690	...	0.713	0.716	0.715	0.707
	ru	0.710	0.746	0.755	0.685	0.746	0.742	0.695	0.694	...	0.774	0.759	0.751	0.735
	pt	0.689	0.732	0.731	0.699	0.729	0.723	0.672	0.670	...	0.735	0.720	0.734	0.715
	ar	0.750	0.751	0.755	0.720	0.754	0.747	0.734	0.761	...	0.726	0.721	0.750	0.740
	de	0.686	0.731	0.736	0.687	0.730	0.731	0.689	0.687	...	0.730	0.722	0.738	0.718
	jp	0.710	0.719	0.715	0.686	0.719	0.710	0.702	0.711	...	0.715	0.698	0.718	0.709
	mk	0.672	0.729	0.733	0.689	0.725	0.721	0.679	0.706	...	0.740	0.731	0.739	0.718
	pl	0.642	0.700	0.712	0.649	0.707	0.702	0.658	0.649	...	0.712	0.719	0.716	0.692
	nl	0.687	0.724	0.726	0.683	0.722	0.715	0.690	0.706	...	0.730	0.713	0.734	0.712

Table 5.3: Macro F1 for the same experiment as above (LaBSE embeddings average of paragraphs and MLP classifier) gives lower numbers in comparison with weighted F1. This shows that there is poor performance in some categories, especially those with a smaller number of articles. The green background shows the best results for each column (each language) and also the best classifier. The trend is the same through almost all experiments: the el classifier is performing very well on other languages, and zht and zhs are very close to each other.

	en	es	fr	mk	pl	nl
Censorship	0.859	0.881	0.853	0.862	0.855	0.801
Development	0.869	0.881	0.879	0.881	0.883	0.881
Digital Activism	0.778	0.811	0.804	0.798	0.795	0.826
Disaster	0.936	0.935	0.937	0.906	0.935	0.924
Economics & Business	0.896	0.897	0.897	0.887	0.905	0.896
Education	0.915	0.915	0.914	0.916	0.928	0.919
Environment	0.914	0.926	0.924	0.904	0.923	0.934
Governance	0.762	0.781	0.788	0.788	0.786	0.821
Health	0.906	0.918	0.898	0.861	0.907	0.905
History	0.889	0.905	0.915	0.874	0.893	0.886
Humanitarian Response	0.918	0.919	0.932	0.898	0.919	0.923
International Relations	0.851	0.867	0.864	0.829	0.853	0.860
Law	0.842	0.835	0.854	0.858	0.851	0.846
Media & Journalism	0.844	0.864	0.845	0.822	0.854	0.837
Migration & Immigration	0.843	0.902	0.884	0.885	0.889	0.889
Politics	0.740	0.744	0.719	0.676	0.746	0.756
Protest	0.816	0.845	0.821	0.812	0.840	0.824
Religion	0.959	0.956	0.959	0.939	0.945	0.955
Sport	0.985	0.987	0.978	0.979	0.985	0.986
Travel	0.912	0.907	0.923	0.914	0.911	0.909
War & Conflict	0.855	0.881	0.832	0.819	0.883	0.890
Technology_Science	0.887	0.899	0.895	0.857	0.886	0.899
WomenGender_LGBTQ+_Youth	0.803	0.836	0.824	0.750	0.829	0.833
Freedom_of_Speech_Human_Rights	0.774	0.760	0.748	0.673	0.746	0.739
Literature_ArtsCulture	0.828	0.859	0.829	0.797	0.850	0.849

Table 5.4: Category-wise F1 for selected languages (the three most numerous and three least numerous languages). This table shows that poor F1 for some categories is not caused by insufficient number of articles for this category. For example Politics category shows worse results than Health, even though there are more articles in Health category.

Weighted F1		TESTED ON												
		en	**es**	**fr**	**mg**	**it**	**el**	**zht**	**zhs**	**...**	**mk**	**pl**	**nl**	**AVG**
TRAINED ON	**en**	0.875	0.875	0.874	0.874	0.872	0.874	0.883	0.892	...	0.880	0.872	0.876	0.875
	es	0.875	0.881	0.879	0.874	0.878	0.880	0.883	0.890	...	0.894	0.882	0.883	0.880
	fr	0.873	0.880	0.880	0.874	0.877	0.879	0.888	0.888	...	0.891	0.879	0.885	0.880
	mg	0.879	0.882	0.882	0.880	0.882	0.883	0.892	0.900	...	0.891	0.881	0.889	0.884
	it	8.874	0.878	0.877	0.874	0.877	0.877	0.886	0.888	...	0.884	0.876	0.885	0.878
	el	0.875	0.883	0.884	0.875	0.883	0.883	0.885	0.893	...	0.889	0.887	0.888	0.884
	zht	0.880	0.882	0.879	0.879	0.881	0.880	0.893	0.935	...	0.882	0.870	0.884	0.882
	zhs	0.873	0.874	0.872	0.874	0.872	0.873	0.897	0.891	...	0.883	0.868	0.879	0.875
	bn	0.864	0.873	0.872	0.868	0.870	0.872	0.878	0.886	...	0.882	0.877	0.879	0.874
	ru	0.868	0.881	0.883	0.870	0.879	0.882	0.882	0.890	...	0.893	0.886	0.889	0.881
	pt	0.862	0.874	0.874	0.872	0.872	0.873	0.881	0.890	...	0.881	0.877	0.882	0.875
	ar	0.875	0.878	0.878	0.871	0.878	0.878	0.890	0.900	...	0.881	0.870	0.885	0.878
	de	0.865	0.877	0.877	0.870	0.877	0.877	0.882	0.887	...	0.888	0.877	0.885	0.877
	jp	0.863	0.867	0.867	0.861	0.867	0.869	0.870	0.878	...	0.884	0.871	0.877	0.870
	mk	0.864	0.876	0.876	0.872	0.873	0.875	0.881	0.894	...	0.887	0.876	0.884	0.877
	pl	0.852	0.870	0.872	0.861	0.870	0.872	0.872	0.876	...	0.881	0.877	0.879	0.870
	nl	0.865	0.875	0.874	0.870	0.872	0.874	0.880	0.892	...	0.888	0.874	0.883	0.875

Table 5.5: This table shows a subset of the results of the experiment when classifiers were only trained on 2000 articles. The results are not significantly worse, which is proof of the high quality of embeddings made with LaBSE.

score in Table X is an average of the results of 10 classifiers. Ten classifiers were trained separately, each of them on a different subset of 2000 articles from the original train/test split.

The conclusion from all the classification experiments is as follows: Most of all, LaBSE and LASER embeddings seem to be of equal quality. They work very well across languages. The best pairs (trained on and tested on) consist of different languages. A simple MLP classifier with two layers is performing well. There are some categories that are classified excellently (F1 $>$ 0.95); on the other hand, some are more difficult to distinguish. In the pre-processing phase, categories of similar topics were merged, but there are still some left that are close in theme.

Chapter 6

Multilingual Topic Discovery

Given embeddings can be used for more NLP tasks, not just classification. One of them is certainly topic discovery. The name of a category can give some information, but it does not tell what the article is about. For this reason, topic discovery, identifying the main themes or subjects discussed in a text corpus, is needed.

This chapter describes different techniques used for the topic discovery task. At first, K-Means clustering is used to form groups of articles. Then the TF-IDF approach was used. It worked well, but it turned out it was not suitable for the web application. It is able to define the most representative words for each cluster, but only for the languages present in the cluster. Unfortunately, not all the languages showed up in all the clusters.

For this reason, another approach was needed. A Bag of Words representation was created for all the languages, and a multilingual document model was trained using the BoW statistics and embeddings. The most representative words for each cluster are computed not by TF-IDF but using the pretrained model and embedding of the cluster. This way, the most representative word for an article can be obtained just from the pretrained model and article embedding, and there is no language limitation.

6.1 K-Means clustering

K-means clustering is a widely used unsupervised machine learning algorithm that divides a dataset into k clusters based on the similarity of the samples. It is a simple and efficient algorithm commonly used in machine learning that divides a dataset into k clusters based on the similarity of the samples. It is a simple and efficient algorithm commonly used. It starts by randomly assigning each observation to a cluster, and then it iteratively improves the assignment by calculating the mean of each cluster and re-assigning observations to the nearest cluster mean. This process continues until the cluster assignments no longer change or a maximum number of iterations is reached.

The K-means algorithm is based on the principle of minimizing the sum of squared distances between the data points and their respective cluster centers. This objective function is called the within-cluster sum of squares (WCSS), and it measures the quality of the clustering solution. The K-means algorithm tries to find the cluster centers that minimize the WCSS.

$$WCSS = \sum_{C_k}^{C_n} \sum_{d_i in C_i}^{d_m} distance(d_i, C_k)^2,$$

where C is a cluster, d is a data point.

One of the main advantages of the K-means algorithm is its simplicity and efficiency. It is relatively easy to implement and can be applied to datasets of any size. Scikit-learn also features MiniBatchKMeans which is faster and reduces the computational cost thanks to not using all the dataset in each iteration.

Usually, the K-means algorithm also provides a straightforward way to interpret the results, as the clusters can be visualized and analyzed. In the case of 768-dimensional embeddings, the visualization is a bit more complicated. It is not always possible to visualize high-dimensional data in 2D space. The t-SNE technique [11] was used to visualize data in a two- or three-dimensional map, however, without success. Clusters are not visually well separated, which is understandable due to the nature of the data. Articles belong to multiple categories that overlap with each other. Clusters can, however, be represented by the most representative words in different languages.

6.2 Topic discovery using TF-IDF

Clusters obtained thanks to MiniBatchKMeans are represented by the center of this cluster, which is again a vector of 768 dimensions. This is impossible to interpret and therefore needs to be done differently. For clusters of articles, the most logical solution is to do a topic discovery, thus discovering the most frequently occurring words in each sequence. Considering a multi-language dataset, it is necessary to take the most frequently occurring words for every language in the given cluster.

The easiest way to do it is to use TF-IDF (Term Frequency-Inverse Document Frequency). It is a numerical statistic that is used to evaluate the importance of a term in a document or a collection of documents. It is commonly used in text mining, information retrieval, and other NLP tasks. The TF-IDF score is calculated based on two factors:

1. The term frequency (TF) measures the number of times a word or a phrase appears in a document and gives an indication of the importance of the term within the document.

2. The inverse document frequency (IDF) measures how common a word or a phrase is across all documents in a corpus, and it helps to identify the words or phrases that are more unique or specific to a particular document. It helps to get rid of stop words (words that do not carry any sense, e.g., articles, prepositions).

The formula for calculating the TF-IDF score for a term in a document (for each word w in a cluster c) is:

$$tfidf(w, c) = tf(w_i, c) * idf(w_i)$$

Particularly when working with clusters:

$$tf(w, c) = \frac{n(w_i, c_k)}{\sum_{j=1}^{W} n(w_j, c_k)},$$

where w = word, c = cluster, n = number, W = size of dictionary (number of words in total), C = number of clusters.

$$idf(w_i) = \log \frac{D_k + 0.01}{d_k(w_i) + 0.001}$$

Language	10 most representative words
en	the, is, in, government, of, police, president, law, on, was
es	el, la, de, gobierno, ley, contra, policía, presidente, elecciones, derechos
fr	le, la, gouvernement, été, loi, président, contre, police, ont, droits
mg	lalàna, mpanao, governemanta, polisy, fifidianana, gazety, politika, filoha, tamin, fitsarana
it	governo, ha, di, polizia, legge, stato, contro, del, presidente, diritti
el	την, της, ότι, στις, και, για, κυβέρνηση, τον, κατά, οι
ru	за, против, по, года, власти, интернет, закон, правительство, facebook, после
pt	de, da, governo, lei, contra, presidente, foi, polícia, direitos, no
de	der, gegen, wurde, des, regierung, die, am, dass, polizei, gesetz
pl	za, że, praw, przeciwko, prezydenta, prawa, człowieka, został, rządu, rząd
nl	de, van, op, tegen, werd, regering, door, hij, dat, politie

Table 6.1: This table shows the most representative words for a particular article.

where D_k = total number of documents in the cluster, $d_k(w_i)$ = number of documents, in which the word occurs.

CountVectorizer from Scikit-learn is able to create a sparse matrix for calculating TF. IDF can be computed straightforwardly according to the Formula 6.2. Then, the 10 most frequently occurring words can be taken as a description of the cluster. Since there are 16 languages, this calculation has to be made for every language present in the cluster. Thus, each cluster is represented by $L * 10$ words, where L is the number of languages in the cluster.

This approach was giving good results, but due to the inability to integrate into the web application described in the introduction of this chapter, it was not used in the end. Instead, a multilingual BoW model was trained.

6.3 Multilingual bag-of-words model for topic discovery

Another possible approach makes use of both prepared embeddings and the texts of the articles. At first, a bag of words (BoW) representation is created from articles, language-wise. Therefore, each document is represented as a vector of word frequencies, ignoring the order and structure of words in the sentences. BoW vectors of all the articles of a given language are stored in one file, which is used by the multilingual BoW model. At first, the vocabulary size was limited to 25 000, but this number turned out to be insufficient; increasing it to 100 000 led to better results, even though none of the present languages reached a vocabulary of this size. Spanish articles have the widest vocabulary with 71,448 words, despite the fact that there are more English articles than Spanish ones.

The model uses BoW representation and document embeddings to learn to represent words from different languages in a shared embedding space. The model therefore learns a word embedding matrix for each language. In the beginning, the bias vector is initialized with the log of the unigram distribution over vocabulary. The model is trained to maximize

the log-likelihood obtained from the known unigram distribution over the documents and the distribution computed using learned parameters. The model is displayed in Figure 6.1.

This way, it is possible to generate the most representative words in all the languages for every article inserted into a web application. There is no K-Means model needed, only the embedding of the document and the pretrained model. The model provides a word embedding matrix, which is multiplied by the embedding of the article, and the top 10 words with the highest score (logits) are chosen as the most representative ones. The example is displayed in Table 6.1.

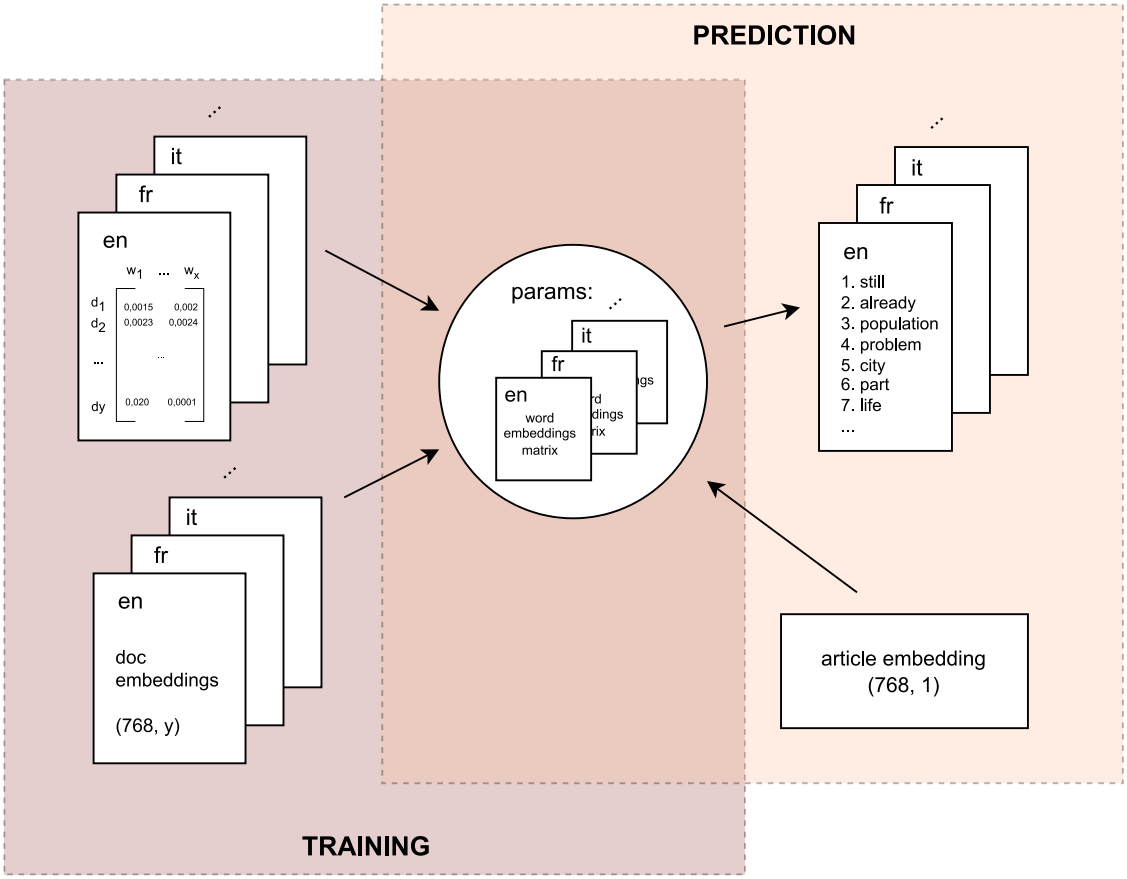

Figure 6.1: From unigram distribution in documents and document embeddings, the model learns the word embedding matrix for each language. Then, in production, it only needs the article embedding to produce the most representative words in all the languages.

Chapter 7

Web application

The classification scores show that both models are providing embeddings of comparable quality, and topic discovery works well. To demonstrate the results achieved, a simple web demo was developed.

Deploying an ML model to a web application is, in general, crucial to making it useful and accessible to a wider audience. A typical ML model requires specialized software and hardware, which makes it challenging for end users and non-technical collaborators to gain trust in ML and provide feedback on model development [1]. If a model is deployed as a web application, it can be accessed from anywhere with an internet connection. Users can easily access it from their desktop or mobile device without having to install any software, in this case, mainly the right version of Python packages. In addition, a web application provides a user interface, which makes it easy for users to interact with the model.

Since the motivation of this thesis is to test multilingual models on a dataset consisting of a wider number of languages than existed before, the web application is not a priority. The goal is not to create a perfect end solution; the web application is intended more as a demo, providing predictions in real-time. For this reason, Gradio, in combination with Hugging Face Spaces for hosting, was used as one of the fastest ways to deploy ML models to a friendly web interface.

7.1 Gradio

Gradio is an open-source Python library that allows developers to quickly create customizable user interfaces for their machine learning models [1]. With Gradio, it is possible to create interactive web-based applications that allow users to interact with the ML model in real-time.

The authors of Gradio were informed by interviews with machine learning researchers, and based on this information, they published the first version of the open-source package Gradio in 2019. The package supports audio, image, and text-based models and includes various user interface components.

The basis of the web application is an interface object. Through the parameters of this object, theme, title, and other properties of the application can be set. It is also used to define the input and output of the application, or more specifically, the components that allow it to receive input and display output.

For the web application created for the purpose of this thesis, input is a textbox where the user can insert text from the article. After submitting it, pretrained models are invoked, as follows:

1. At first, an embedding is created. For this, the LaBSE model is used. It was chosen because both LASER and LaBSE models gave similar results, and LaBSE is easy to import from Hugging Face. The strategy of creating an average embedding from the whole is applied since it gives the best results for classification. An alternative would be to create an embedding just from the title of the article, but this is not suitable for a topic discovery task. The title is too short and does not contain much information (in many words). The embedding is used just for backend computations; it is not displayed to the end user.

2. The second task is classification. The embedding is forwarded to a pre-trained classifier trained on an English corpus. This is, however, just a matter of preference because all the classifiers give similar scores, no matter the language of the training corpus. In any case, the output of this model is a list of probabilities for all the possible classes. This output is displayed to the user. It is shown in the form of the names of categories and a bar chart. Categories names are written in English, regardless of the language of the article; this convention was adopted from the beginning because GlobalVoices categories are not perfectly unified through languages. A bar chart visualizes the probability of categories; therefore, it allows one to see if the model was sure of its decision or if there are some other minority categories. The threshold is set to 0.5.

3. Then, the embedding obtained in step 1 is used for the topic discovery task. A multilingual document model and files with bag of words stats for each language are needed. The top 10 words describing the article are obtained by multiplying the word embedding matrix with the article embedding. Then those scores are ordered, and words with the highest scores are chosen from the bag of words files.

As written above, the Gradio application has one input and three outputs. The first is the list of categories; the second is the bar chart; and the third is the list of most representative words. The interface is divided into two tabs, with classification in the first one and topic discovery in the other.

The web application architecture is demonstrated in Figure 7.1, whereas the interface can be seen in Figure 7.2.

7.2 Hugging Face

Hugging Face is a technology company that specializes in natural language processing (NLP) and develops software tools to help people work with language data. It was established in 2016 by French entrepreneurs, and it immediately gained popularity mainly for its open-source software library, Transformers, which has become a standard tool in the NLP industry.

Transformers is a Python library that provides pre-trained models for various NLP tasks, such as sentiment analysis, language generation, and machine translation [24]. The library uses a transformer architecture, which is described in Chapter 3. Transformers make it easy to use and fine-tune pre-trained models, which are suitable for various applications like chatbots, language understanding systems, or machine translations. After all, even this work uses the LaBSE model from the Transformers library.

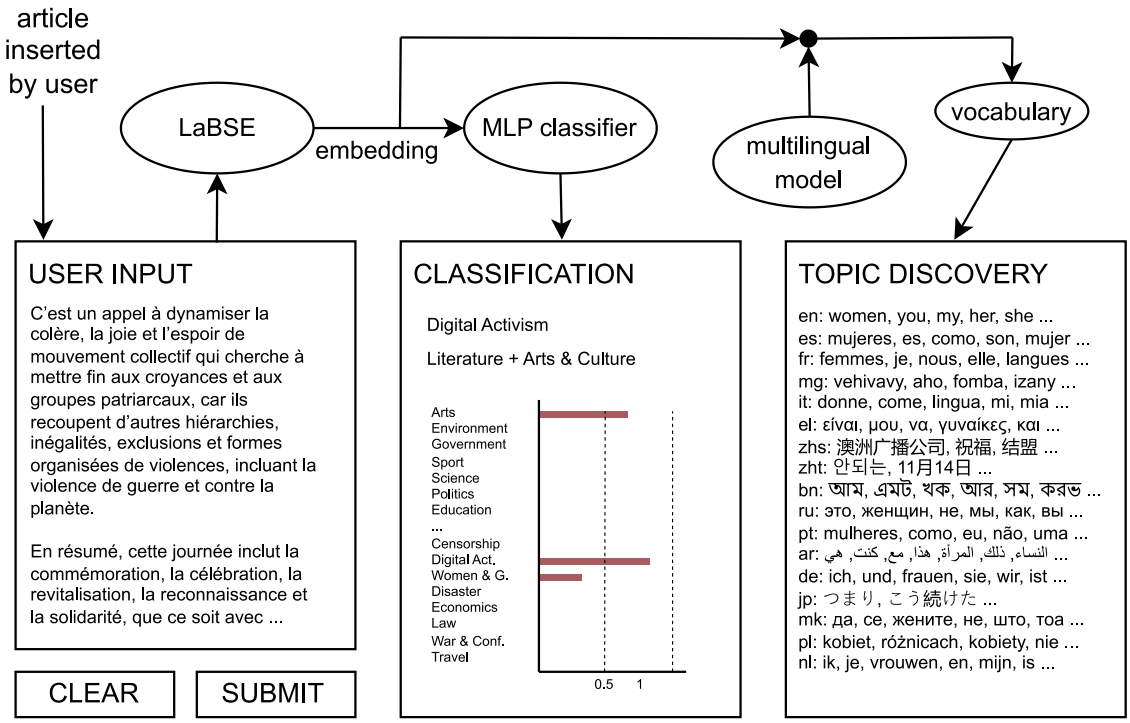

Figure 7.1: The diagram shows the web application architecture. The user inserts the text of the article, and after background computations, classification results and the most representative words are displayed to the user.

Besides this, Hugging Face has developed other tools and products, such as Tokenizers, a library for efficient text tokenization, and Datasets, a collection of datasets for NLP tasks. For this thesis, the most relevant tool from Hugging Face is, apart from the Transformers library, a platform called Hugging Face Hub. This platform brings together the NLP community and allows them to discover and share NLP models, datasets, and scripts. The platform also includes Spaces.

Hugging Face Hub provides a central repository where users can upload and download NLP resources, and it is integrated with Spaces. With Spaces, it is possible to create a workspace and customize it. Different templates for common NLP tasks like question answering or text classification are prepared. The workspace can then be shared with collaborators. Code is run here in a cloud-based environment in a containerized environment with pre-installed dependencies, and users can choose from a variety of hardware configurations. Above all this, Spaces can be paired with Gradio.

While creating a brand new Space, Gradio can be chosen as the SDK. Then Space will automatically initialize itself with the latest version of Gradio, and the user can continue with work. Hugging Face Spaces are Git repositories; therefore, files are uploaded by pushing commits. Hardware properties are set in settings, and Python libraries are specified in requirements.txt. The dependencies are installed automatically after pushing the files. There is also a web interface for uploading or even creating files. In any case, working locally, where Gradio can run on a localhost, and then pushing files is much more convenient. Each Space environment provides 16 GB of RAM, 2 CPU cores, and 50GB of disk space in the free version. Although the web application is running a bit slower, the free setup is sufficient for demo purposes.

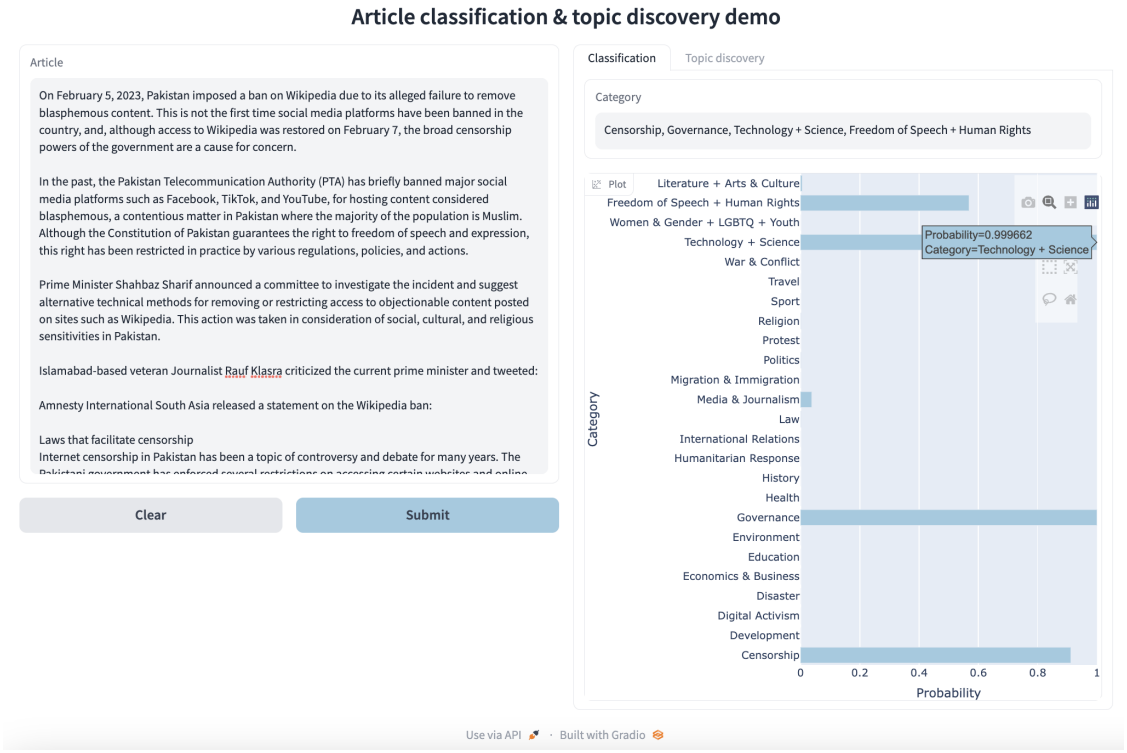

Figure 7.2: A user inserted into the web application an article about Pakistan banning Wikipedia. It is correctly classified into *Censorship*, *Governance*, *Technology + Science* and *Freedom of Speech + Human Rights* categories.

The web app runs on `https://huggingface.co/spaces/andufkova/articles`.

Chapter 8

Conclusion

The goal of this thesis was to create a multilingual dataset and benchmark existing pretrained language models using this dataset. There were no previously existing datasets diverse enough to be suitable for multilingual and multilabel article classification. A dataset of almost 250,000 articles in dozens of languages was created. For this task, different scraping libraries and various news sites were considered. In the end, the dataset is based on the GlobalVoices.org website, which provides news articles translated into different languages. Some information was scraped with the News-please library; some of it was scraped manually. Minor errors were discovered in the functionality of the library mentioned before, like mistakes in language codes.

A dataset of this size needs to be well organized. The way it is done has changed several times as new knowledge has come to light. The articles are stored in folders language-wise; for each one, both structured JSON and original HTML versions are stored, since from HTML code it is possible to extract further information, which may be useful in following up on this work. Information about categories is stored separately in metafiles. A CSV file stores categories for each article, and JSON file stores information about which translations of articles belong together.

The original dataset was filtered. Some categories were merged, which led to 25 categories for the final dataset. Only 17 languages with more than 2000 articles were preserved, and the concerned articles were encoded as document embeddings. Two models were chosen for this task: LaBSE and LASER. The original intention was to use LASER3, but it turned out that version 3 just added new languages, mainly low-resource languages not presented in the created dataset; therefore, LASER2 is used. As LaBSE is implemented in the HuggingFace library, it was more straight-forward to obtain document embeddings, and for this reason, initial experiments started with this model. Different strategies for embedding creation were adopted.

The purpose of multilingual models is to allow training on high-resource languages and predicting on low-resource languages, so the aim of this thesis was to train on one language and test on others. The dataset was strictly split into train and test sets by year, even though it was not ideal for some languages. It was important that the article in one language not appear in the train set and its translation not appear in the test set. This way, zero-shot learning could be performed.

For the classification task, the initial experiments were performed with MLP, SVM, and Logistic Regression. MLP gave slightly better results, so the following experiments were conducted with this classifier. The best results were obtained with embeddings obtained as an average embedding of all the paragraphs. Both LaBSE and LASER2 produced com-

parable results, with weighted F1 varying from approximately 0.82 to 0.89. Comparison with macro F1 showed that there is poor performance in some categories, the ones with a smaller number of articles or the ones that are not so strictly separable. The best news is that F1 scores were similar no matter which language the model was trained and tested on; therefore, the results of zero-shot learning were as good as the results of training and testing on the same language. An experiment with 2,000 training articles per language was performed, and the results were still the same. All of this shows the quality of multilingual embeddings.

With the created embeddings, a topic discovery task was performed. The first idea was to do K-Means clustering and, with TF-IDF, get the most representative words for every cluster. Then a new article would have been assigned to a cluster and represented by significant words for that cluster. This approach was implemented and worked well, but it turned out to not be very suitable for the final web application. In addition, the words representing the whole cluster were often too general. Due to this, a multilingual bag-of-words model was trained. From document embeddings and unigram distribution over vocabulary, the model learned a word embedding matrix for every language. To obtain the most representative words, only article embedding is needed. It is multiplied by the word embedding matrix, and words with the highest score are taken for every language.

Finally, both tasks, classification and topic discovery, are demonstrated in a simple web application created with Gradio and hosted on Hugging Face Spaces. It is available on `https://huggingface.co/spaces/andufkova/articles`. The user can insert the text of the article, document embedding is created in the backend, and it is proceeded to the classifier and multilingual model for topic discovery. The predicted categories and the most representative words are then displayed to the user.

To summarize, multilingual models showed good performance. Both tested models gave comparable results, even with a low number of training examples. For future work, it would be interesting to try more low-resource languages and merge the GlobalVoices dataset with other data (even the existing ones). This would require mapping the topic labels and further modifications, yet it would allow for a more general dataset.

Bibliography

[1] Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A. et al. Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. *CoRR*. 2019, abs/1906.02569. Available at: http://arxiv.org/abs/1906.02569.

[2] Artetxe, M. and Schwenk, H. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *CoRR*. 2018, abs/1812.10464. Available at: http://arxiv.org/abs/1812.10464.

[3] Chang, M.-W., Ratinov, L., Roth, D. and Srikumar, V. Importance of Semantic Representation: Dataless Classification. In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*. AAAI Press, 2008, p. 830–835. AAAI'08. ISBN 9781577353683.

[4] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, p. 4171–4186. DOI: 10.18653/v1/N19-1423. Available at: https://aclanthology.org/N19-1423.

[5] Feng, F., Yang, Y., Cer, D., Arivazhagan, N. and Wang, W. Language-agnostic BERT Sentence Embedding. *CoRR*. 2020, abs/2007.01852. Available at: https://arxiv.org/abs/2007.01852.

[6] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Hamborg, F., Meuschke, N., Breitinger, C. and Gipp, B. News-please: A Generic News Crawler and Extractor. In: *Proceedings of the 15th International Symposium of Information Science*. March 2017, p. 218–223. DOI: 10.5281/zenodo.4120316.

[8] Heffernan, K., Çelebi, O. and Schwenk, H. Bitext Mining Using Distilled Sentence Representations for Low-Resource Languages. In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, December 2022, p. 2101–2112. Available at: https://aclanthology.org/2022.findings-emnlp.154.

[9] Lample, G. and Conneau, A. Cross-lingual Language Model Pretraining. *CoRR*. 2019, abs/1901.07291. Available at: http://arxiv.org/abs/1901.07291.

[10] Lewis, D., Yang, Y., Russell Rose, T. and Li, F. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*. april 2004, vol. 5, p. 361–397.

[11] Maaten, L. van der and Hinton, G. Visualizing Data using t-SNE. *Journal of Machine Learning Research*. 2008, vol. 9, p. 2579–2605. Available at: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[12] Mikolov, T., Yih, W.-t. and Zweig, G. Linguistic regularities in continuous space word representations. In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013, p. 746–751.

[13] Mitchell, R. *Web Scraping with Python: Collecting More Data from the Modern Web*. O'Reilly Media, 2018. ISBN 9781491985526.

[14] Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020.

[15] NLLB Team, Costa jussà, M. R., Cross, J., Çelebi, O., Elbayad, M. et al. *No Language Left Behind: Scaling Human-Centered Machine Translation*. arXiv, 2022. DOI: 10.48550/ARXIV.2207.04672. Available at: https://arxiv.org/abs/2207.04672.

[16] Palen Michel, C., Kim, J. and Lignos, C. Multilingual Open Text Release 1: Public Domain News in 44 Languages. In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, June 2022, p. 2080–2089. Available at: https://aclanthology.org/2022.lrec-1.224.

[17] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, p. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[18] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, vol. 12, p. 2825–2830.

[19] Read, J., Pfahringer, B., Holmes, G. and Frank, E. Classifier Chains for Multi-label Classification. In:. August 2009, vol. 85, p. 254–269. DOI: 10.1007/978-3-642-04174-7_17. ISBN 978-3-642-04173-0.

[20] Reimers, N. and Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR*. 2019, abs/1908.10084. Available at: http://arxiv.org/abs/1908.10084.

[21] Richardson, L. Beautiful soup documentation. *April*. 2007.

[22] Shuyo, N. *Language Detection Library for Java*. 2010. Available at: http://code.google.com/p/language-detection/.

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is All you Need. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30. Available at: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[24] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C. et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR*. 2019, abs/1910.03771. Available at: http://arxiv.org/abs/1910.03771.

[25] Zhang, M.-L. and Zhou, Z.-H. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Transactions on Knowledge and Data Engineering*. 2006, vol. 18, no. 10, p. 1338–1351. DOI: 10.1109/TKDE.2006.162.

Appendix A

Languages included in dataset

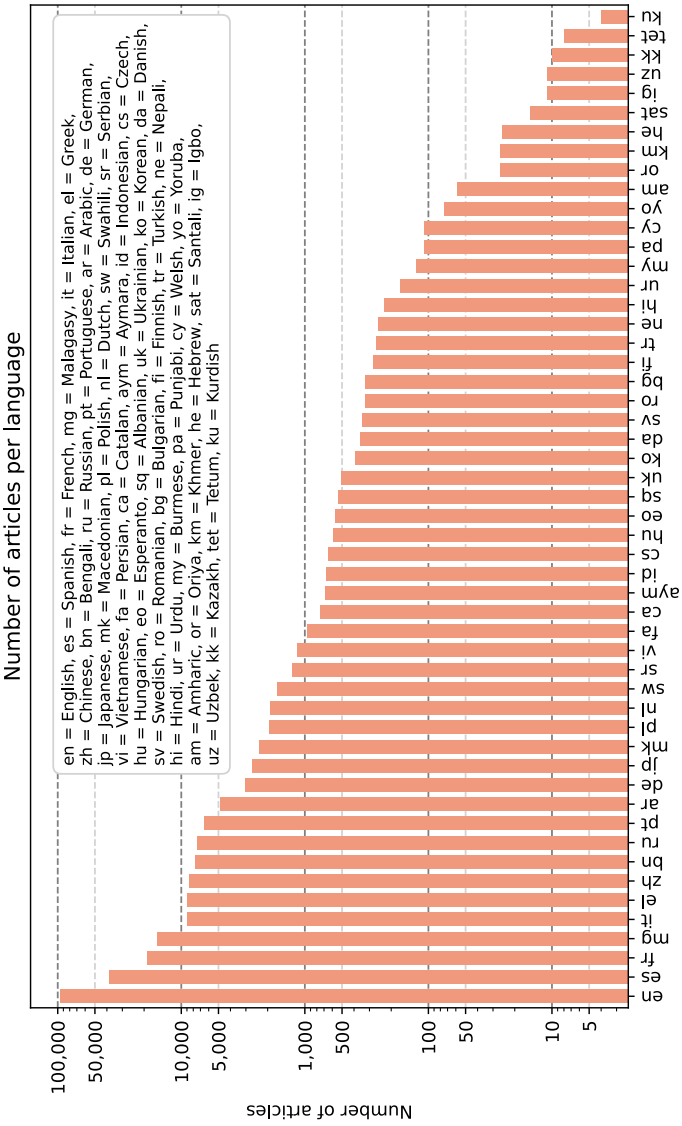

Figure A.1: The image shows number of downloaded articles per language. English is main language with almost 100 000 articles out of 245 821 overall. The least numerous languages were Kurdish (ku) with 4 articles, Tetum (tet) with 8 articles and Kazakh (kk) with 10 articles.

Appendix B

MLP classifier + LaBSE results

Weighted F1	TESTED ON																				AVG
TRAINED ON	en	es	fr	mg	it	el	zht	zhs	bn	ru	pt	ar	de	jp	mk	pl	nl				
en	0.884	0.883	0.881	0.893	0.882	0.885	0.895	0.908	0.863	0.878	0.882	0.889	0.896	0.880	0.898	0.882	0.885				0.886
es	0.881	0.886	0.885	0.886	0.885	0.887	0.888	0.898	0.862	0.881	0.889	0.887	0.896	0.883	0.901	0.887	0.889				0.887
fr	0.874	0.883	0.884	0.884	0.882	0.884	0.885	0.900	0.853	0.881	0.888	0.886	0.898	0.878	0.893	0.890	0.892				0.884
mg	0.884	0.885	0.884	0.882	0.884	0.886	0.893	0.906	0.867	0.882	0.886	0.890	0.896	0.881	0.899	0.883	0.888				0.887
it	0.874	0.881	0.881	0.879	0.880	0.882	0.882	0.890	0.849	0.877	0.885	0.883	0.894	0.876	0.892	0.877	0.884				0.880
el	0.826	0.839	0.835	0.841	0.837	0.833	0.849	0.859	0.821	0.834	0.843	0.840	0.854	0.847	0.852	0.841	0.844				0.841
zht	0.880	0.881	0.878	0.879	0.880	0.880	0.894	0.933	0.845	0.871	0.878	0.888	0.890	0.871	0.883	0.870	0.882				0.882
zhs	0.873	0.874	0.872	0.875	0.873	0.873	0.898	0.893	0.843	0.866	0.876	0.882	0.885	0.868	0.883	0.869	0.879				0.875
bn	0.865	0.873	0.872	0.868	0.870	0.873	0.878	0.886	0.859	0.871	0.878	0.877	0.888	0.872	0.880	0.876	0.879				0.874
ru	0.868	0.886	0.889	0.872	0.884	0.887	0.881	0.887	0.851	0.888	0.896	0.886	0.908	0.882	0.907	0.893	0.894				0.886
pt	0.869	0.880	0.880	0.879	0.878	0.880	0.881	0.888	0.850	0.877	0.886	0.883	0.891	0.872	0.885	0.877	0.886				0.879
ar	0.883	0.886	0.887	0.878	0.886	0.886	0.896	0.912	0.862	0.879	0.888	0.889	0.898	0.878	0.887	0.880	0.892				0.886
de	0.868	0.879	0.880	0.872	0.878	0.881	0.884	0.888	0.860	0.875	0.884	0.881	0.891	0.873	0.889	0.878	0.888				0.879
jp	0.863	0.867	0.867	0.861	0.868	0.869	0.869	0.880	0.848	0.867	0.874	0.868	0.882	0.875	0.883	0.871	0.877				0.870
mk	0.826	0.841	0.839	0.839	0.839	0.836	0.852	0.858	0.831	0.840	0.848	0.845	0.855	0.851	0.854	0.840	0.849				0.844
pl	0.855	0.863	0.861	0.848	0.861	0.857	0.865	0.869	0.851	0.862	0.869	0.855	0.873	0.867	0.871	0.864	0.869				0.862
nl	0.822	0.833	0.834	0.834	0.832	0.830	0.853	0.862	0.822	0.829	0.839	0.836	0.848	0.847	0.843	0.839	0.839				0.838

Table B.1: This table shows weighted F1 scores of classification performed with LaBSE embeddings (the technique taking average of paragraphs) and MLP classifier. All the languages presented in dataset give very similar results despite different number of articles for each language. The numbers on diagonal represent training and testing on the same language.