Semester project: Language-independent websites embeddings

Sylvain Lugeon
Master student / EPFL
sylvain.lugeon@epfl.ch

Tiziano Piccardi
Supervisor / EPFL
tiziano.piccardi@epfl.ch

Abstract

Websites embeddings can be used for various downstream tasks, in particular topics classification. Current methods for computing embeddings. only work for English websites, making it an issue since the web is a very multilingual space. We propose a language-independent embedding based on both visual and textual features, making use of XLM-R for handling multiple languages. We assess the capability of the proposed embedding on a multilingual classification task using samples from DMOZ directory. Results show an average accuracy of 0.83 and zero-shot accuracy of 0.70 when the evaluation is done in a language not seen during training.

1 Introduction

There exists more than 1.7 billion websites today, that is twice the number of websites there were 5 years ago (Statista, 2019). Internet is growing exponentially. We therefore need tools that enable a machine to automatically compare, analysis or classify websites. Internet is also a very multilingual platform: 40% of the websites are not available in English (W3Techs, 2020).

The aim of this project is to find a language-independant vector representation for websites, a.k.a *websites embeddings*. In a first step we describe the different features used for constructing the embedding. We use both textual-based and visual-based features. In a second step, we evaluate the embedding on a classification task. The code that supports this report can be found on github.com/epfl-dlab/websites_embedding.

2 Related work

The idea to represent websites with vectors and evaluate them on a classification task isn't new. In 2009, a study from EPFL try to use only urls to classify websites (Baykan et al., 2009). Using n-grams and Support Vector Machine, they are able to get a F-1 score of 82.4 on the DMOZ dataset (see section 3.2). A more recent study from Zhejiang University uses not only the text content of a website, but also the HTML tags structure to get an accuracy of 95.7 on sub-datasets from DMOZ (Deng et al., 2020). So far, all studies focus on English websites and there is no published work about multilingual websites classification.

Important signals are contained in the textual elements representing a website: its url, its content or its HTML file. These signals can be extracted using NLP techniques and models (n-grams, word2vec, GloVe, etc.). In our case we need a multilingual NLP model. For this project, we rely exhaustively on (pre-trained) XLM-R (Conneau et al., 2020), which achieves state-of-the-art performances in cross-lingual understanding tasks. In particular, when a sequence of characters (a sentence, a word) is fed to XLM-R, it is tokenized and a vector of 768 dimensions is returned for each token. We then get the *sequence embedding* by averaging the token's vectors.

3 Method

3.1 Questions

We want our embedding to perform well for a various number of tasks, especially for classification. The first question we want to answer is the following:

1 How well does our website embedding perform in a multilingual classification task?

English is the dominant language on Internet. Datasets about labelled websites often have entries in a limited number of languages, if not exclusively in English. In a machine learning task, this can make the training particularly sensitive as we are

only able to train our model from samples in a few number of languages. The second question we want to answer is therefore the following:

2 Does our model have zero-shot properties with languages not seen during the training?

Finally, the computation of certain features may be expensive. We make the choice to use computationally expensive models that involve computer vision or complex deep model (XLM-R). For someone who want to use these embeddings in practice, maybe that the use of such features is not worth the few percentages they add up to the classification task. The last question we want to answer is:

3 What is the impact of the different features on the prediction performance?

We'll answer the first question by jointly fitting a classifier with samples of different languages. The second question will be tackled by evaluating the trained classifier with samples coming exclusively from a language not seen during the training. Finally, we'll run a ablation study to answer the third question.

3.2 Dataset

The data in this project exclusively come from DMOZ (previously known as Open Directory Project), an open-content directory of the web powered by AOL. The dataset we use is a 2016 record of DMOZ's entries and is publicly available on Harvard's data portal (Sood, 2016). The websites are classified using a hierarchical ontologies and are of different languages. There are more than 2.5 millions entries in the dataset, partitioned in 14 categories. The use of this dataset is motivated by two facts (certainly not independent): most of the researches about websites classifications also use DMOZ, which will make us able to compare our results with others, plus there is no real choice apart DMOZ when it comes to a public, multilingual, labelled set of websites.

As the names in the ontologies are different for each languages, a manual matching sas to be done to group the websites under the same category names. For example, the category corresponding to *Recreation* in the English ontology is called *Hobbies* in the French ontology. The category corresponding the *Regional* has to be dropped, since there is no equivalence in others ontologies than English. The number of categories becomes therefore 13.

We draw 3 different, non-overlapping, sets of samples from the DMOZ dataset.

1. An **experiment** set, used to assess the performances of the final classifier. Size: 136'980 samples.
2. A **visual** set, used to train the computer vision model (CVM). Size: 218,569 samples.
3. A **zero-shot** set, used to assess how well the classifier performs with a language non-seen during the training phase. Size: 10,400 samples.

We need two different set for the classification and visual parts because, as the CVM will be used to extract visual features, it is trained independently and of the classifier.

For the experiment set, we use websites from four different languages, that are part of the most represented languages in DMOZ: English, German, French and Italian. The zero-shot test is composed of Japanese websites and the visual set of English websites only.

The data can be unbalanced in two ways: the number of websites in each languages can vary, as well as the numbers of websites in each category. We decide to balance the experiment set only with respect to the languages, this is motivated by the fact the multilingualism is in the core of our problem. We create our experiment set by selecting the maximum number of samples so that for each category, the samples are uniformly distributed among the four selected languages. Figure 1 shows the distribution in the categories for the experiment set.

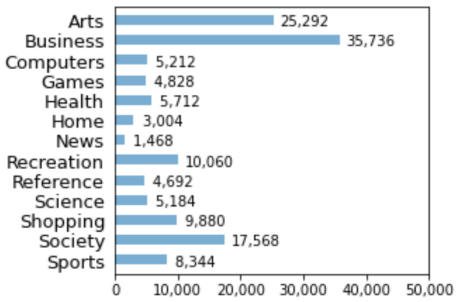

Figure 1: Number of samples per category in the experiment set

Ideally, the visual set would have need to be composed of websites from different languages. But we don't have enough samples to balance both

the experiment and visual set. We then choose to train the computer vision model exclusively with English samples and select up to 20,000 samples from each category, if available. The distribution is showed on figure 2

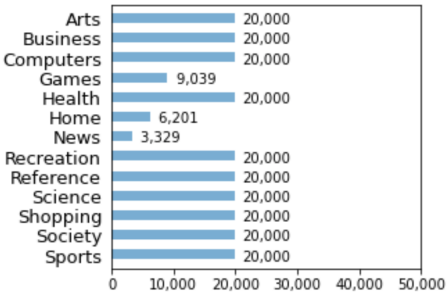

Figure 2: Number of samples per category in the visual set

The zero-shot test is balanced with respect to the category, 800 samples are taken from each category.

A website can contain various sub-pages. The embedding of a website is only computed from its *homepage*.

3.3 Tools

All code is done in Python. We make use of handy additional libraries, in particular: *Sentence-Transformers* (Reimers and Gurevych, 2020) to efficiently compute sentence embeddings with pre-trained XLM-R, *PyTorch* (Paszke et al., 2019) for implementing all machine or deep learning modules, *BeautifulSoup* (Richardson, 2007) to extract elements from HTML files and *Selenium* to capture screenshots of websites.

Training the CVM and the classifiers is done on a GeForce GTX TITAN X with 12Go of memory.

4 Features

Table 1 displays the features used to create the embedding, as well as their types and dimensions.

4.1 Url

The top level domain (TLD) of a website can be an indicator about its content. A website ending with *.org* would most likely be about a non-profit organization, while a website ending with *.edu* would certainly relate to education. We keep the 17 most frequent, non-country code (like *.ch* or *.us*), TLDs in the dataset and create embeddings in a one-hot fashion. The retained TLDs are: *com, org, net, edu, info, biz, gov, tv, me, name, coop, pro, fm, int, aero,*

Feature	type	dimension
TLD	one-hot	17
Url	xlmr	768
Text content	xlmr	768
Metatags	one-hot	20
Description	xlmr	768
Keywords	xlmr	768
Title	xlmr	768
Links	xlmr	768
Visual	resnet18	512

Table 1: Summary of the features used to create websites embeddings.

club, church. We note that this feature may convey only little information for non-english website.

The sub-domains can also be informative. Others researches showed that applying n-grams to the url led to good prediction performances. We decide to rather rely on XLM-R to extract meaningful groups of characters (i.e words) in the url and their semantic. Table 2 shows how XLM-R tokenizer can separate the words for facitious urls in multiple languages. A pleasing consequence of using XLM-R, as oppose to n-grams, is that we do not need to train an extra model for each language to cover. Urls are first cleared of the scheme (http://), world wide web prefix (www.) and dashes (- and _). Then, only the lowest sub-domain of the url is fed to XLM-R and we (simplistically) call it *url feature*. The decision to keep only the lowest sub-domain is because higher sub-domains may contain information, but not within the semantic (e.g second-level domain *.co* that indicates a commercial content, but has no semantic value).

Facetious url	Tokenization
booksshopping	books, shop, ping
livresachat	livres, achat
comprarelibri	comprar, e, libri
bücherkaufen	bücher, kauf, en

Table 2: XLM-R tokenizer applied to urls in English, French, Italian and German.

4.2 Text content

When we think of the task of classifying websites, the first feature that pops up in our mind may be the textual content. We tokenize the displayed text at the sentence-level, feed each sentence to XLM-R and average the outputs to create a *text content*

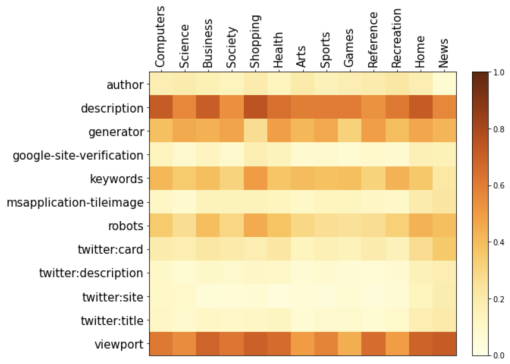

Figure 3: Fraction of websites that have a given metatag field per categories.

feature.

4.3 Metatags

The ⟨meta⟩ tags in the HTML source code contains a lot of information under various fields: the *description* field acts like a summary of the website's content, the *keywords* field is composed of terms that inform search engines about the content, finally the *title* is what is shown on a search engine's result page or web browser's window when displaying the website. We'll feed each of this three fields to XLM-R to create three distinct features. In a same manner that for the text content, the description is tokenized at the sentence level, fed to XLM-R and averaged on the outputs. The keywords and title are fed to XLM-R without pre-tokenization. Additionally, the title has been cleaned of special characters (e.g * or :).

Metatags can contain several fields, others than the ones used below. Without looking at the content of a field, just its presence can be informative about the content of the website. Figure 3 displays a heatmap of the presence of certain fields, with respect to the categories. We remark some variance within some fields, meaning that there may be indicative about the content. If a website contains a twitter-related metatag, then it would probably be about *Home* or *News*. On the other hand, a website with a *generator* metatag would less likely be about *Shopping*. We keep the 20 metatags with the most variance (of the presence) and use one-hot vectors, which we call *metatags feature*.

4.4 Links

A homepage may possess links to others pages, we particularly think here of pages under the same domain. The url of these links can be useful when inferring the topic of the homepage. As an example, here are the 5 first links from the EPFL homepage (*www.epfl.ch*):

- www.epfl.ch/fr/
- www.epfl.ch/about/fr/a-propos/
- www.epfl.ch/education/fr/education/
- www.epfl.ch/research/fr/recherche/
- www.epfl.ch/innovation/fr/innovation-2/

Even if we have no clue about EPFL or its homepage, we can guess that it must be a school or university homepage when looking at the words in the urls below. We extract all the links in the homepage, split them into words, feed the 10 most frequent words to XLM-R and average their outputs. To follow-up the example above, the 10 most frequent words contained in the links from EPFL homepage are: *fr, research, education, campus, schools, services, domains, de, phd, et*.

Keeping only the most frequent words is motivated by the fact that there exists a lot of redundancy in the links (some only differs in their last part). This also prevent the complexity of the embedding to explode, as the number of links for some websites can be huge (more than a thousand) but some convey only little information.

4.5 Visual features

We think that, as humans, if we are shown a screenshot of a website with the text having blurred, we'll be able to make an educated guess about its topic. The structure of the webpage, the colors used or the images can tell us a lot about the website's content. Another motivation for using visual content is that it is less dependant on the language than textual content, if not at all.

We first train a computer vision model (CVM) to classify websites. Then, when given a website to embed, we fed-forward its screenshot through the trained model and use the values from the second-to-last layer as a *visual feature*.

5 Computer vision model

5.1 Architecture

The goal of the CVM is to extract patterns from the screenshot of a website. We start from the pre-trained version of the ResNet-18 (He et al., 2016) model and replace the last dense layer with a layer that matches the number of classes for our problem (13). This model has been trained for a classification task on the ImageNet dataset (Russakovsky et al., 2015), a collection of images of objects

coming from 1000 classes. ResNet-18 is the simplest model presented in the aforementioned paper among more complex models, based on the same architecture, that were considered state-of-the-art for image recognition. The use of this model is motivated by its capacity-to-complexity ratio, it remains a simple model to train while showing good performances for image recognition tasks. The total number of parameters is 11,183,181.

5.2 Training procedure

The visual set is randomly divided into a training (80%) and testing (20%) sets. First we take a screenshot of the website, as if it was open in a browser window of size 1,920x1,800 pixels. Then we downsample the image by a factor 3, to get a resolution of 640x360 pixels. Inspiring ourselves from the ResNet paper, we use the following techniques to avoid overfitting.

Data augmentation. Training samples are randomly cropped with a ratio of 0.6, hue and saturation are jittered by a factor 0.2. To fed images of the same size to the network, a 5-crop transformation (five corners + center) of the same ratio is applied to testing samples and their respective outputs is averaged.

Weighted loss. Classes that are underrepresented have their loss lifted by a constant. The weights used are inversely proportional to the representation of a class in the training dataset.

Adaptive learning rate. Starting from 1e-4, the learning rate is decreases by a factor 0.1 when the testing accuracy is not increasing for the past five epochs.

We train the model for 100 epochs, figure 4 shows the history of the training and testing accuracies, as well as the losses. We note than the testing loss increases while the testing accuracy also increases, this is due the cross entropy loss that penalizes false predictions more strongly than rewarding true predictions.

5.3 Training result

The final testing accuracy is **0.324**. Figure 5 plots the confusion matrix of the testing set. We note that some classes are more easily identified (e.g *Computers*, *Arts* or *Sports*) while others are more often misclassified (e.g *Home*). Moreover, we can remark some classes proximity in the misclassified samples: samples with true labels *Game* or *Science* are often predicted as *Computers*, or samples

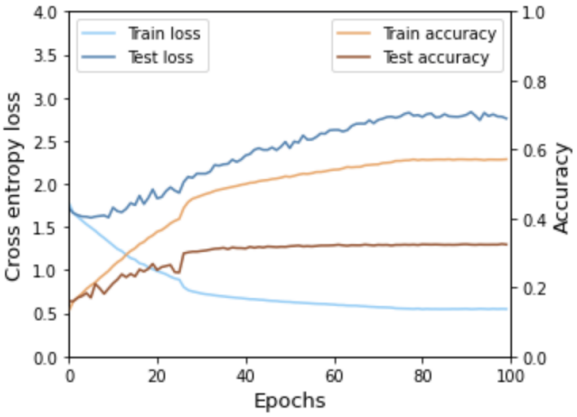

Figure 4: Training history of the computer vision model.

coming from the *Home* category can sometimes be predicted as *Shopping*.

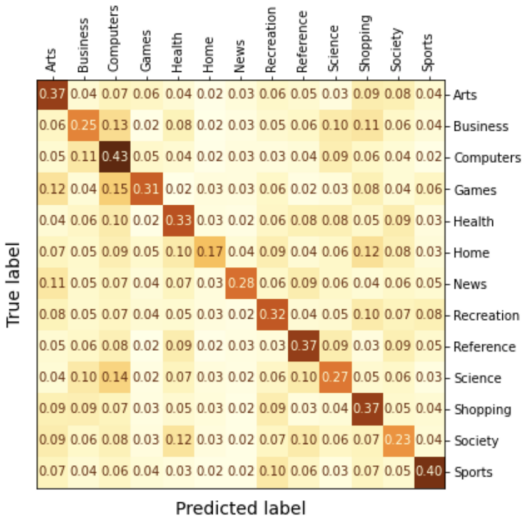

Figure 5: Confusion matrix of the visual testing set. Normalization is done with respect to true labels.

6 Pre-processing

6.1 Erronneus urls

Dealing with websites is a source of noise. Websites can be closed, moved or simply not updated. Moreover, DMOZ is a community-edited dataset and may contain broken or inaccurate urls, as well as urls returning files that are not webpages (e.g to download a document). The three sets defined in section 3.2 are first cleaned of potentially *erronneus* urls before any computation or analysis. We follow a simple policy: we discard websites that does not return a 200 code status or whose content is not an HTML file.

This implies that approximately 20% of the urls are discarded, and that the number of samples on which is conducted the project is slightly lower than the ones presented in section 3.2. We control that the discarded samples are more or less distributed uniformly among all classes and languages.

We also make sure that all samples are *homepages* and not *sub-pages*. Actually, all samples from DMOZ should be homepages. As a sanity check, we verify that no slash character (/) lie inside the urls.

6.2 Missing features

Some features presented in section 4 may be missing for some websites. The concerned features are the ones extracted from the HTML file, namely: *text content, description, keywords, title, links*. The purpose of all these textual features, in addition to the *url* feature that is never missing, is to capture the semantic using XLM-R. We assume that the semantic of these features should not differ by a lot, and therefore choose to replace a missing textual feature by the average of the others present textual features.

Table 3 highlights the fraction of samples in the experiment set that are missing a given feature. We note that a lot of websites don't have a *description* or *keywords* fields in their metatags.

Feature	Missing
Text content	2.1%
Description	41.1%
Keywords	68.7%
Title	3.1%
Links	17.7%

Table 3: Percentage of samples that miss a given feature in the experiment set.

7 Classification task

We train a binary classifier for each class, with the task to separate positive and negative samples. For a given class, we select negative samples uniformly among the others classes so that the number of positive and negative samples are the same.

Because the numbers of positive samples is limited for certain classes, we use 5-fold cross validation and report the average recall, precision and accuracy, alongside with their 95% confidence intervals computed on the folds scores.

We use the same setup for all experiments. The (selected) features are concatenated and the classifier is a simple dense layer followed by a sigmoid function (a.k.a logistic regression). We use dropout of probability 0.5 to prevent overfitting. The loss is binary cross entropy.

The training phase consist of 100 epochs. We start with a learning rate of 1e-4 and decrease it by a factor 0.1 when the test accuracy plateaus for 10 epochs, in a similar manner than with the computer vision model.

8 Experiments

8.1 Multilingual classification

We use the experiment set for training and testing. The performances of each binary classifier are shown in figure 6. The average precision, recall and accuracy are respectively **0.841**, **0.814** and **0.830**. The fraction of correctly predicted test samples per languages is shown in figure 7, with 95% confidence intervals.

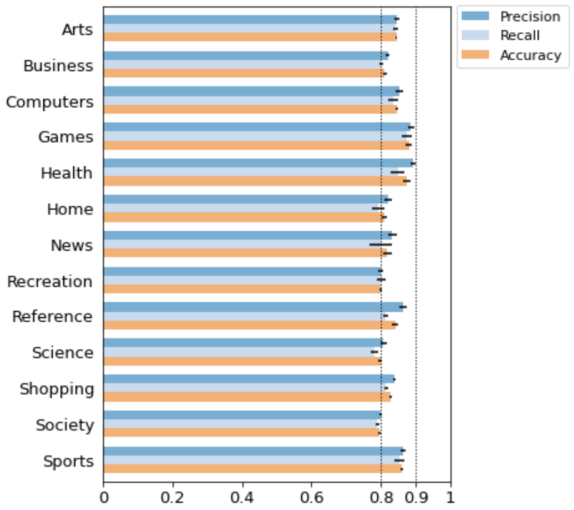

Figure 6: Test scores of each binary classifier, samples come from 4 languages, all features are selected.

8.2 Zero-shot classification

We keep the same 5-fold setup but instead of testing the classifier on the unused fold from the experiment set, we test it on positive and negative samples from the zero-shot set. The results are shown in figure 8. The average precision, recall and accuracy are **0.779**, **0.574** and **0.703**.

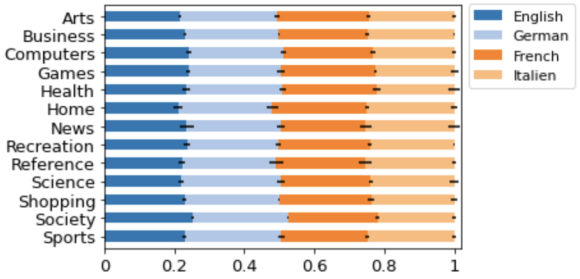

Figure 7: Fraction of correctly predicted test samples per language, confidence intervals apply to the quantity directly at their left.

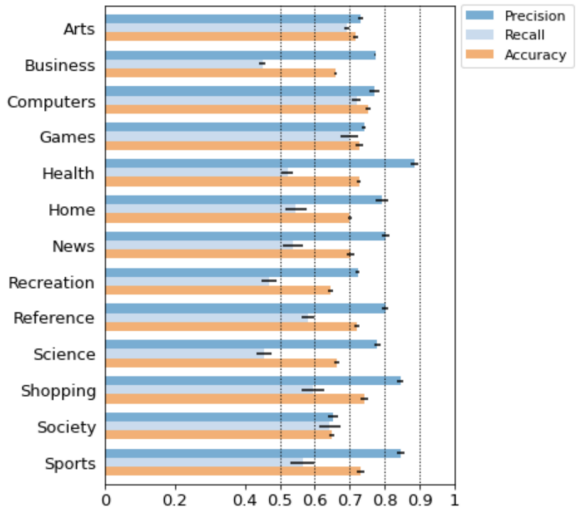

Figure 8: Test scores with a language not present in the training set (Japanese).

8.3 Performance vs. complexity

We gradually remove features in the order someone with limited resources would do. We define the following group of features, that goes from simplest to the most complex:

1. **Offline:** *TLD* and *url* features. This is the most basic set of features, that does not even require to access the website.

2. **Meta:** *Metatags*, *title*, *description*, *keywords*, *links* features. This set requires to fetch the HTML file and a moderate use of XLM-R, as the number of sentences to embed is limited.

3. **Text:** *Text content* feature, this require a most exhaustive use of XLM-R as we fed-forward all the textual content of the website.

4. **Visual:** *Visual* feature, in addition to fetching the HTML file, we also need to capture a screenshot of the website.

Figure 9 plots the average classification scores, depending on the most complex set of features used. For a certain level of complexity, all features equally or less complex have been used. For example a classification at the *meta* level indicates that we use features of the *meta* group as well as of the *offline* group. For each fold, we compute the average across the categories and report the 95% confidence intervals.

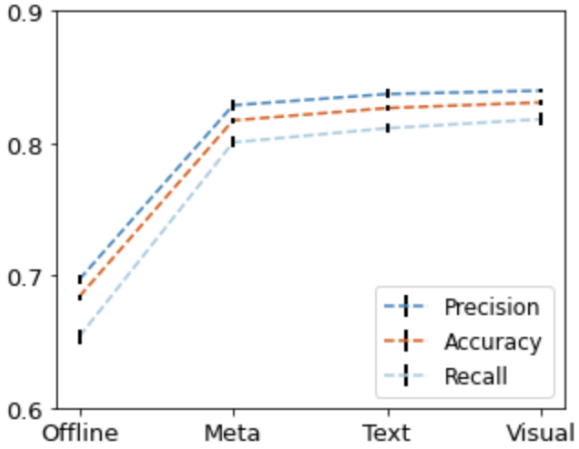

Figure 9: Average scores versus complexity, the features for each complexity level are described in section 8.3.

9 Discussion

With accuracies between 0.8 and 0.9 depending on the class, we can assess that the constructed *websites embeddings* accurately capture the content of the website. Some classes are more identifiable than others, but all have an accuracy above 0.8. Moreover, using XLM-R allows to create language-independent embeddings. Looking at figure 7, we see that prediction correctness does not depend on the language.

Thanks to this, a classifier trained on four European languages can perform well for samples in Japanese, with a mean precision of almost 0.8 and which can go up to 0.9 for some classes. The classification is nevertheless less accurate than when the training and testing are done on the same languages. The recall is particularly low, for same classes even worse than a random policy. Looking at figures 6, 8 and 9, we remark than the recall is lower than the precision in all experiments.

The predictive power of some features is sometimes not worth the cost. From figure 9, we can

see that classification based only on the urls and metatags can perform well, with an average accuracy of 0.817. Thus, doing the extra step of fed-forwarding all the text content to XLM-R, taking a screenshot of the website and fed-forwarding it to a deep model only increase the accuracy of +1.3%. We do not think that no information lies in the text content or the screenshot, but that the information may already be present in others, less complex, features.

Concerning the CVM, we are surprisingly pleased that it can capture so much signal, from every class. Whether it captures the visual structure of the website, visual content like images, or maybe even font size is hard to say and would require further analysis.

Finally, comparing our work with previous studies, we do not succeed to push the classification performances. However, the aim of this project was different in its essence: we wanted an embedding to perform well for all languages, whereas all previous studies only focus on monolingual (English) classification. We can also note that because we needed to kept the languages balances, we used a smaller training set than studies presented in section 2 (by a magnitude 10 compared to the first study).

The validity of the dataset itself can also be questioned. DMOZ is a community-edited directory and we have no guarantee of its exactness. An inspection of its content shows that a non-negligible number of the websites are old or do not represent well the assigned category. This is supported by the fact that 20% of the urls could not be accessed or weren't HTML files. The fraction of missing *description* or *keywords* fields presented in table 3 also indicates a partial incompleteness of some websites: predictions were only made from a limit set of features for a great fraction of samples. A perfect score is therefore hard to obtain on such a dataset.

10 Further work

The analysis should be repeated on others datasets, such as Alexa's top sites by categories or Wikipedia's external links. These datasets may be of better quality than DMOZ. To compare our performances with others studies in a pertinent way, we could evaluate our embedding on a English-only setup. The CVM could be further analysed to understand the form of the signal it captures. In a last step, we could release our trained classifiers in the shape of an public API.

References

Eda Baykan, Monika Henzinger, Ludmila Marian, and Ingmar Weber. 2009. Purely url-based topic classification. pages 1109–1110.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

L. Deng, Xin Du, and J. Shen. 2020. Web page classification based on heterogeneous features and a combination of multiple classifiers. *Frontiers of Information Technology Electronic Engineering*, 21:1004 – 995.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Leonard Richardson. 2007. Beautiful soup documentation. *April*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. Imagenet large scale visual recognition challenge.

Gaurav Sood. 2016. Parsed DMOZ data.