Interreg CENTRAL EUROPE
ProsperAMnet
European Union
European Regional
Development Fund

FACTSHEET OF ARTIFICIAL INTELLIGENT PROGRAMMING (MACHINE LEARNING MODELS; NATURAL LANGUAGE PROCESSING ALGORITHMS)

D.T2.3.1

Version 1
03 2021

A. Table of Contents

This document is intended to provide an overview regarding some of the experiments conducted for the implementation of the Service Export Radar.

B. Company similarity

In this task, we have to find the most similar companies to a given company.

1. Dataset

For our preliminary experiments, we downloaded all websites based on the manufacturing part of a USA company database. The downloading process was limited to 200 pages and 2 levels of depth.

We extracted the following categories of textual information from the webpages:

- *text*: The visible texts from the webpage.
- *title*: The title of the webpage.
- *keywords*: The content of the keywords html meta tag.
- *description*: The content of the description html meta tag.
- *h123*: The content of the 1st, 2nd and 3rd level headings.
- *h123456*: The content of all headings.
- *concat*: The combination of all previous fields.

The evaluation of the different methods was based on the SIC codes of the companies.

1.1. Refining and cleaning of the dataset

We applied different types of filters to make this dataset as clean as possible. The most common issue we found was the problem of the domain selling websites. These sites once belonged to a company, but currently, these are just empty pages that only contain a link where you can buy them. An approach that was based on the usage of regular expressions was utilized to remove the broken and domain selling websites. This method marks pages based on some predefined expressions like “domain has been suspended” and automatically searches similar sites to them.

We also dropped those companies with more than one sic code and the ones where we could not extract one of the textual categories (*title*, *keywords*, ...).

The final dataset contains 5653 companies.

2. Representation of the websites

The word embeddings represent each word with a vector. The main idea behind these embeddings is if the meaning of two words is close to each other, then the vector of these words will be close too. Based on these embeddings we created a vector representation for each website for each text category.

One of the main advantages of these embeddings for us is the possibility to handle the multi-linguality. For this reason, we used the aligned fastText[1] multi-lingual embeddings. To get a representation for a website we calculated the vector of each word and got the mean of them.

[1] https://fasttext.cc/docs/en/aligned-vectors.html

3. Evaluation

Based on the vector representation of the websites we calculated the most similar *k* companies to each company. The first two digits of the SIC codes were applied as the labels of the document. To compare the results, we calculated the precision @k (P@k) and mean average precision @k measures (MAP@k).

3.1. Source of information

In these experiments, we are comparing which parts of an html document could help the most. We use all of the downloaded webpages from each website.

	P@1	P@5	MAP@5
text	0.4778	0.4301	0.5590
title	0.4350	0.3921	0.5178
description	0.4396	0.3982	0.5245
keywords	**0.4916**	**0.4512**	**0.5723**
h123	0.3758	0.3352	0.4607
h123456	0.3791	0.3378	0.4647
concat	0.4702	0.4291	0.5555

We get the best results by using the *keywords* meta tag. This tag contains the main focuses of a company without a huge amount of irrelevant information. In the case of professional websites, this tag works best, but only ~20% of the companies fulfill this information. The visible textual content, what a human can see reaches the second place.

3.2. Is the main page enough?

The dropping of the subpages could make a huge impact on the speed of the system. In the second round, we checked what happens if we keep only the main page of websites.

		P@1	P@5	MAP@5
text	whole website	0.4778	0.4301	0.5590
	main page	0.4415	0.4029	0.5300
keywords	whole website	**0.4916**	0.4512	0.5723
	main page	0.4840	**0.4581**	**0.5857**

In the case of *keywords*, there is only a small difference between the two systems and in two of three metrics, the main page based solution was better. In the case of the *text* category, there is a larger gap, the scores got lower when we dropped the subpages. A potential reason for this is that the main pages usually contain the most important keywords, but in some cases the textual parts do not contain enough details.

3.3. Application of keyword extraction

The usage of keywords makes the best results but in ~80% of the companies, we have not got this information. One of the most obvious answers for this problem is the keyword extraction, but in that case we have to handle the problem of multilinguality, and we need a solution for each language. Instead of that, we would like to exploit the multilingual possibilities of our embedded vector representations. Our method maps the vector space of the visible texts to the vector space of the keywords. For this experiment, we split the data into a train (80%) and a test (20%) set. We added 20% of the companies where we did not find keywords to the test. Finally, we got 4,686 companies.

On the train set we trained a regressor that maps the vectors of visible texts to the vectors of keywords. To get these results we used a linear regressor. (We made experiments with other regressors like neural networks, but these experiments did not deliver better results.) The table shows the results of the following methods:

- text: uses the visible texts from the webpage.
- original + mapped keywords: if there is a keywords tag then we use the content of that, otherwise the mapped vector.
- mapped keywords: we used the mapped vectors for each company.

	P@1	P@5	MAP@5
text	0.4123	0.3711	0.4967
original + mapped keywords	0.4725	0.4377	0.5522
mapped keywords	**0.4733**	**0.4434**	**0.5524**

In conclusion, our mapping solution reached the level of the original keyword field.

4. Multilinguality

Like we mentioned earlier one of the main advantages of this embedding-based setup is the handling of multilinguality. In a multilingual embedding, the English word *ale* will be close to the German word *Bier*. In the previous experiments, we evaluated our systems on an English-only dataset. When we started to manually check what happens in multilingual cases, we found an unexpected behavior.

In most cases when we calculated the most similar companies to a given one the language of the webpages was the same. E.g.: if we searched similar pages to a German page most of the result was German too.

While the similarity calculation works fine in word level between languages, in the case of website level the language dependent specificities make a large bias.

4.1. Dataset

Because the previous dataset was specialized to US companies, we moved to the database of Europages. We did the same downloading and filtering process and selected 100,000 randomly sampled documents from that. The languages, ordered by their decreasing order of frequency, ended up as shown in the below table.

ISO language code	Frequency
en	36,909
de	36,704
it	14,983
fr	10,017
cs	1,997
hu	770
sk	485
sl	148

4.2. Method

The Europages provides English keywords for many companies. By using these keywords we made an any language to English mapping instead of using the previous English to English mapping.

4.3. Evaluation

We selected 5000-5000 English, German, Italian and French websites to the test set and made the train set from the others. The category system of Europages was used as labels.

To measure the different systems in multilingual setup we checked some extra statistics.

- *Prop. diff. lang*: the proportion of the guesses where the language of the query company and the language of the most similar company are different.
- *Same lang P@1*: Precision@1 where the language of the query company and the language of the most similar company are the same.
- *Diff. lang P@1*: Precision@1 where the language of the query company and the language of the most similar company are different.

	P@1	Prop. diff. lang	Same lang P@1	Diff. lang P@1
w/o mapping	0.3019	0.0002	0.3019	0.0000
with mapping	**0.3452**	**0.2622**	**0.3671**	**0.2836**

C. Service classification

In this other task, we have to find the services which are provided by a given company. Based on our annotations we created a classifier that assigns service categories to the webpages of a company. There are 21 different service categories that derive from 6 main categories.

5. Dataset

There is a dataset where the website of the companies was annotated manually. The annotations had two rounds, after the first general annotation the second round was focused on the rare categories where we had not got enough examples in the first round.

To get a larger and more diverse dataset we downloaded the websites of the companies from the first annotation round. The downloading process was limited to 500 pages and 3 levels of depths for each website.

From the second annotation round, each page is counted as a unique website. We did similar clearing processes that were mentioned in the case of company similarity.

5.1. On the case of “false false positive" predictions

In order to simplify and speed up the job of the annotators, the annotators were allowed to mark only the first occurrence of the services for every domain. As the annotators could finish with the annotation of a single page faster, they were able to check more companies. As such, this decision resulted in a trade-off between the depth and the width of the annotations process.

As a consequence, it is then possible that the entire website of some company contains multiple evidence for that company offering a certain kind of service, yet, our annotation procedure did not manage to find all those pages that can support that given service. This means that the machine learning problem we faced, cannot be considered as a standard supervised learning problem, but more as a problem where learning had to be performed from positive and unlabeled instances (with the unlabeled instances belonging to either of the positive or the negative class).

A machine learning model performs a false positive prediction when it says that an input belongs to a certain (positive) class, however, the annotated gold standard dataset contains the opposite information. As described in the previous paragraphs, the gold standard annotations cannot be considered complete for any company. This also implies that some of our predictions regarded as false positive errors are in fact incorrectly treated as such, i.e., they only seem to be false positive, because an annotator failed to find the given website which should otherwise belong to one of the positive classes. In that sense, the false positive rate our models produce are overly pessimistic and should be treated with a grain of salt, for which reason we treated recall a more important measure during our experiments.

In order to reduce the effect of the “false false positive” predictions during evaluation, we dropped all of the pages where the language of the page was not matched with the language of the annotation (e.g.: if the company website andritz.com was annotated in English, we excluded all of the non-English pages from the given URL). We did so, as it was often the case that the same service was mentioned in multiple languages over the various webpages of a company, and the annotators most of the times annotated the presence of some service in a single language. If we had not removed all those websites that did not match with the language of the annotation, we would have risked to include a lot of webpages with corrupted (false) labels in our dataset.

After performing the above-described filtering step, the sizes of the final dataset were[2]:

# webpages	84157
# annotated webpages	2004
# websites	1407

6. Evaluation

This annotated database creates a multilabel classification problem. We analyzed each webpage of a company and aggregated the results at website level. Each number that we provide is the micro average service level result.

6.1. Bag-of-words

The bag-of-words is the most classic machine learning technique to create a vector representation for a document. It creates independent features from each word or word n-gram. One of the main problems is the size of the websites, most of the modern, deep learning-based solutions (e.g.: BERT) are designed for shorter texts like sentences. Instead of that, the bag-of-words models do not deeply depend on the size of the input. And it is much less expensive in the case of computational capacity. On the other hand, the bag-of-words models have no information other than the shape of the words, and the handling of multilinguality could be more problematic.

For these experiments, we used logistic regression with tf-idf weighting, we tried different algorithms, neural networks, random forest, or naive Bayes, but we could not get better results. Furthermore, we dropped each feature that occurred in less than 5 documents.

6.1.1. Applying class weights

The dataset is very imbalanced, there are more non-annotated pages than annotated. To make the pages with service information more important we set the weights of the classes inversely proportional to class frequencies.

	precision	recall	f1
w/o class weights	**0.7743**	0.2652	0.3950
with class weights	0.5937	**0.4839**	**0.5332**

The table shows that the setting of frequency-based class weights can improve the results.

6.1.2. N-grams

An n-gram is a contiguous sequence of n words in a given document. The application of n > 1 n-grams helps to use multiword expressions like “product demonstration”, it could

[2] The number of exact annotations was slightly higher, with the difference originating from the fact that some of the annotated websites was not accessible by the time we tried to crawl them.

improve the results but on the other hand, it makes more features which could slower the prediction.

ngram	precision	recall	f1	# features
1	**0.5937**	0.4839	0.5332	66828
2	0.5090	**0.6258**	0.5614	671630
3	0.5208	0.6093	**0.5616**	1504953

6.1.3. Bagging

The bagging is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms. We applied bagging on the n=3 n-gram model.

	precision	recall	f1
w/o bagging	0.5208	**0.6093**	**0.5616**
with bagging	**0.6136**	0.4473	0.5174

The bagging made our model more precise, but it decreased the recall, for which reason we opted for not applying bagging in the final architecture.

6.1.4. Multilinguality with bag-of-words

By using multilingual word embedding we calculated the most similar *k* English words to the words of the non-English webpages.

We tried three different methods to get the most similar English words:

Aligned fastText: the embeddings that we used in the case of company similarity. The pre-trained vectors computed on Wikipedia using fastText. The alignments are performed with the RCSLS method.

Word2word[3]: another word translation approach. It was trained on the OpenSubtitle dataset.

Phrases + VecMap: the above solutions can not utilize phrases. By using a company based website that we downloaded we extracted phrases, created embeddings over these phrased texts, and aligned these embeddings with VecMap[4].

[3] https://pypi.org/project/word2word/

[4] https://github.com/artetxem/vecmap

	k				
	1	3	5	7	10
aligned fasttext	**0.4863**	**0.4829**	0.4787	0.4754	0.4720
word2word	0.4674	0.4652	0.4658	0.4620	0.4591
phrases + vec map	0.4837	0.4820	**0.4831**	**0.4811**	**0.4779**

We got similar results in the case of the aligned fastText and the phrases + vec map models. Both of them outperformed the word2word system.

6.2. Results using embeddings

We also tried the same embedding based vector representations that we applied in the company similarity section. We created a representation for each webpage by averaging the vectors of the words. We also used the aligned fastText multilingual embeddings. This system produced the following scores in the case of the English dataset:

	precision	recall	f1
Fasttext	0.5137	0.4240	0.4645