Lab 2 Questions and Answers

Guillaume CARRIERE, Paul GROLIER,
Cloé ESCUDIER, Youri COLERA

Contents

1 Language detection

1.1 Question 1.0

If the wrong source language is used on google translate, the website will detect the correct source language and propose to use it. This means in addition to translating the given sentence, google translate also runs the sentence on a classifier to assign the most probable language to it.

1.2 Question 1.1

The distribution is equal, there is the exact same number of sentences for all of the 22 languages in the dataset. We can see that this dataset does not have a lot of languages (only 22), but the languages are very varied with languages from all around the world, some being popular languages (english, chinese), and some being less popular (Urdu, Pushto).

1.3 Question 1.2

For the pre-processing, we chose to keep stop words as they depend on the language, and so could help us. We did not use stemming or lemmatizers as these could transform two different words from different languages into the same word. However, we lowered the text and removed punctuation as it does not add enough information that we could use to detect language.

1.4 Question 1.3

We run the risk of being highly proficient at recognizing languages which are overpresent in our dataset, but be not very good at recognizing languages which are not represented as much in our dataset, as we would not have enough information on these languages to be sufficiently trained.

1.5 Question 1.4

Depending on the model, you can give more weight to samples from under-represented languages during training so that the model learns the different languages with equal efficiency. Ultimately, it's possible to find additional data to compensate under-represented languages, or even remove sentences from over represented languages if the training is really impacted.

1.6 Question 1.5

To provide a basic baseline for language detection, we will use a Multinomial Naive Bayes classifier. This is a simple probabilistic text classifier that works by performing a statistical analysis on the dataset (in bag-of-word representation) in order to compute the probability of documents to be of a certain class by inspecting the frequency of its words. Our metrics will be the accuracy and the confusion matrix, to estimate what are the problematic languages. The overall accuracy is 96% which is pretty good. However, the tokenization of chinese and japanese is not perfect so it could be improved. From the confusion matrix on fig.1 we can see our model has a little trouble with arabic and thai (index 0 and 19). We can also see that sometimes, languages are misinterpreted as english (index 3), which may be caused to the overpresence of english culture in the world (leading to english words sometimes used in non-english languages).

1.7 Question 1.6

When testing with the Tatoeba dataset, we obtained an accuracy of 96 %, which is pretty good.

1.8 Question 1.7

The performance is not as good when testing with the dataset from question 1.5, as we obtained an accuracy of 80 %. We can see from the confusion matrix on fig.2 that our fasttext model has problem with specific languages : pushto (13) which it identifies as ottoman turkish, chinese (1) which it identifies as japanese (8), and thai (19) which it often identifies with different other languages.

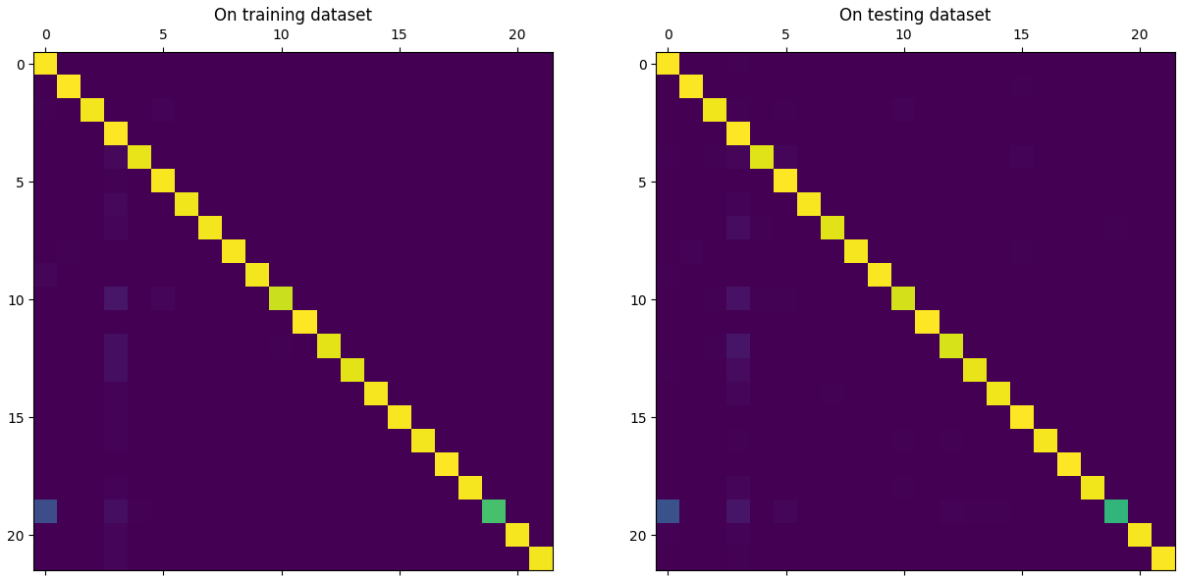

Figure 1: Confusion matrix obtained with the MultinomialNB on both the training and testing dataset

This may be because of difference between the dataset from question 1.5 and the Tatoeba dataset. For example, the Tatoeba dataset contains much more languages than this dataset, which leads our trained model to often classify a sentence with a language that is not present in the dataset. The tatoeba dataset also mostly contains short sentences, while our dataset sometimes has long paragraphs of multiple sentences (especially true for chinese).

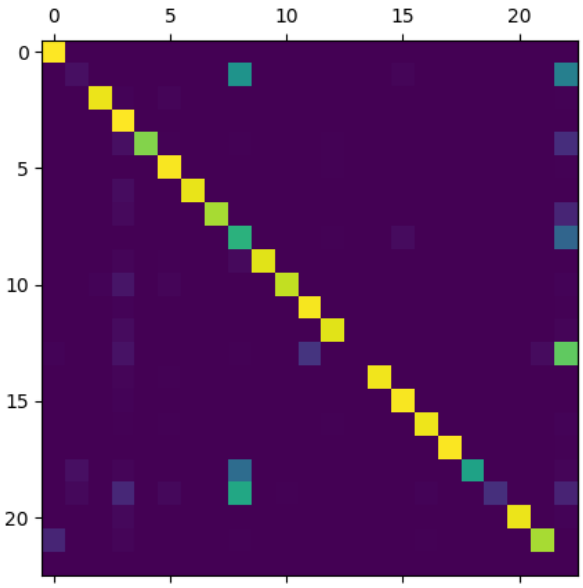

Figure 2: Confusion matrix obtained with fasttext on the testing dataset (Index 22 corresponds to when fasttext outputs a language that is not present on the dataset)

1.9 Question 1.8

We computed precision and recall for a given class. When testing with japanese, we found a precision of 31.5% and a recall of 63% for both our implementation and sklearn's implementation.

1.10 Question 1.9

To improve the performance of our fasttext model we could try to clean its dataset with a custom preprocessing function. We could also improve this dataset by adding other languages that may not be present. Finally, we could perform hyperparameter tuning on the fastText training process, train on more epochs or set a special update strategy on the learning rate.

1.11 Question 1.10

It depends on the context, if we need to work on only a few languages, with a dataset limited in size, then a probabilistic model like MultionmialNB is enough to achieve good performance. With a lot of languages and a big dataset, we would choose fasttext as it can learn on a way bigger dataset than a simple MultinomialNB classifier, which allows for better performance and being able to detect a multitude of different languages with one model.

1.12 Question 1.11

Our strategy would be to detect tokens which are quoted (as in words between " or « characters), and deduct them from N_1 and N_2 depending on their language. This is because we consider that if a language is only used in quotations, it may not be the language of the whole sentence. Then, the language attributed will be the one that holds the most remaining tokens (in our case, english if $N_1 > N_2$ and french otherwise).

1.13 Question 1.12

Generally, a multilingual architecture is robust to multiple languages, as it has knowledge over different languages that could allow for an abstract understanding of the sentence. For example one could imagine an architecture that embeds each word depending on its source language using multilingual language detection. This however depends on context : in a task such as identifying the language of a whole sentence, you have to ultimately make a choice of only one language even if the sentence contains several languages you can understand.

2 Rotate two semantic spaces

2.1 Question 2.1

The idea of MUSE is to provide a library for learning a mapping from a source embedding representing a language to a target embedding representing another language. In a supervised setting, MUSE has a ground truth, in the form of a dictionary that can convert any word from the first language to the corresponding one in the second language, as well as the embeddings of both languages. To learn, it will compute possible mappings from the first embedding to the second, create dictionaries using these mappings, and compare the learned dictionaries to the ground truth dictionary to determine and save the best mapping. The way MUSE computes mapping is by solving the Orthogonal Procrustes problem, where we want to find an orthogonal matrix R that most efficiently maps a matrix A to a matrix B (in our case A and B will be the embeddings). This is done by computing the matrix $M = BA^T$, and then, using Singular Value Decomposition (SVD), compute the nearest orthogonal matrix of M, which is proven to be the same as R.

2.2 Question 2.2

One limit of this approach is that the alignment will depend on a ground truth dictionary which may not be perfect, as it can be hard to construct, and its size will necessarily be fixed, and potentially too small. This limit is not present on unsupervised approaches as it does not rely on a dictionary.

2.3 Question 2.3

It could be possible to align two semantic spaces from a domain specific field by learning the mapping using monolingual corporas containing data only related to this specific field. In the supervised setting,

this could be further improved by also focusing the vocabulary of the ground truth dictionary on the vocabulary of the specific field.

2.4 Question 2.4

We chose to do this by computing a mapping from the french space to the English space using the unsupervised method of the MUSE library. It works by computing a good mapping with an adversarial method, i.e. train a discriminator to differentiate between the mapped source embeddings and the target embeddings and learn a mapping that can fool the discriminator. The resulting mapping is also refined through iterative Procrustes in the same fashion as the MUSE supervised method. The embeddings that we will use to represent the french and English space will be the fasttext Wikipedia embeddings.

2.5 Question 2.5

The mapping is visualized on fig.3 by reducing the dimensionality of the embeddings from 300 to 2 using truncated SVD. The results do not show the best performance (English words may be far apart from their mapped french counterpart) but that might be due to the SVD. However, we can see that the English and French space without the mapping are clearly separated while the mapping merges the french space with the English one. This means we can at least see that the rotation of the french space probably worked.

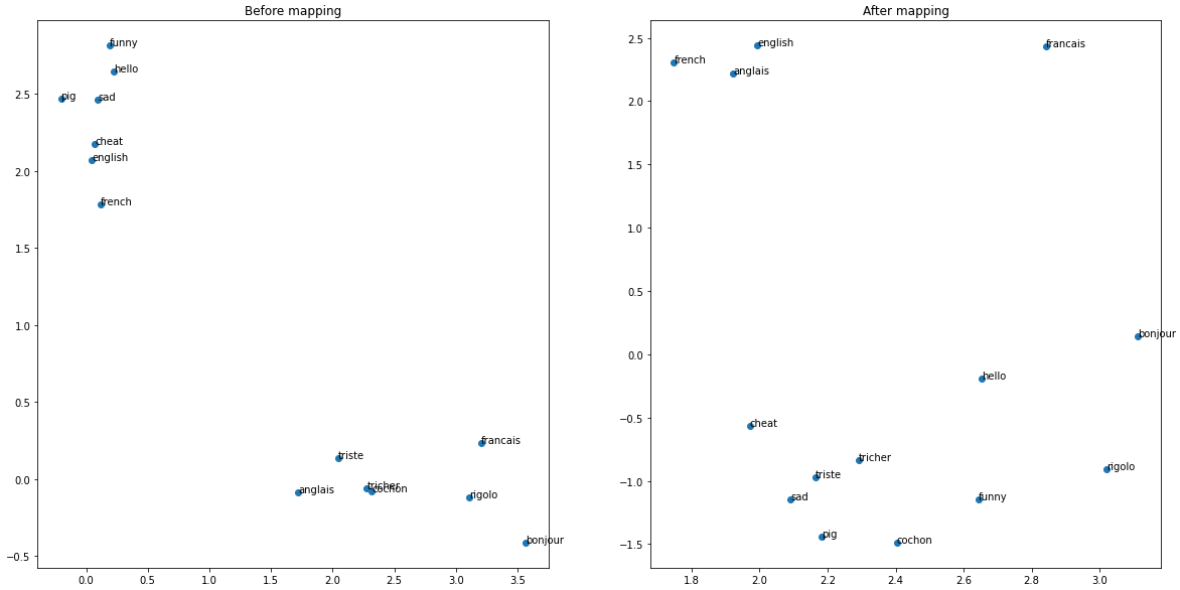

Figure 3: Reduced word embeddings of a few examples before and after mapping

2.6 Question 2.6

To find the translation of a French word in English, first we must apply the learned mapping to the embedding of the french word. Then we can go through the English embeddings and find the embedding closest to the mapped embedding of the french word. The result will be the English word corresponding to this found embedding. To measure how close embeddings are, we will be using cosine similarity as it is the similarity measurement that was used during the learning of the mapping.

2.7 Question 2.7

We can see that the resulting translations on fig.4 are mostly accurate. This approach falls short on unusual words, that probably don't appear often on our dataset, such as "ornythorinque" in our example.

```
cannelle : cinnamon
lapin : rabbit
ornithorynque : monotreme
changer : changing
squelette : skeleton
```

Figure 4: Translations of a few examples using our approach

2.8 Question 2.8

The limit is that only singular words are aligned, which means the grammar is lost when translating, leading to a poor translation. To improve on this, maybe it could be possible to learn alignments on n-grams instead of words to have more meaningful translations.

2.9 Question 2.9

1. Use a trained model (like fasttext) to identify the language of the given sentences $\{s_1, ..., s_N\}$
2. Use previously learned mappings aligning different languages to English to compute embeddings of the sentence's words depending on their language. Preferably, the mappings could work on n-grams, as previously mentioned, to efficiently map the sentences.
3. Use an LSTM architecture previously trained on sentiment analysis with an English corpus (and the embedding used for the mappings) to classify the resulting embeddings of step 2.

2.10 Question 2.10

The first step would be to find at least a corpus C_1 of English data labeled for sentiment analysis related to the specific field, as well as a corpus C_2 of multilingual data also related to the specific field. Once that is done, we need to train a model (like fasttext) to identify the language of a sentence using C_2, learn mappings from the languages of C_2 to English (with unsupervised MUSE for example), and finally train a model to perform sentiment analysis using C_1 as dataset. Finally, we can follow the procedure described in Question 2.9 to perform sentiment analysis on sentences from multiple language using our models and mappings, as these will be specifically designed for sentences related to our specific field.

3 Attention Exploration

3.1 Question 3.a

For the value of c to be closest to a value vector v_j, the corresponding attention weight a_j must be closest to 1 while the others attention weights must be closer to 0, whatever the value vectors. For example, this can be achieved by having of vector of ones for the q and the k_j vector, while the other k vectors are full of 0, but any setting that results in the aforementioned disposition of attention weights would work.

3.2 Question 3.b

To obtain such result, we need to have the attention weights a_a and a_b closest to $\frac{1}{2}$ and the other attention weights closest to 0 such that we have $c = \frac{1}{2} * v_a + \frac{1}{2} * v_b = \frac{1}{2}(v_a + v_b)$ This can be achieved by using q such that :

$$q = N * (k_a + k_b)$$

With N an arbitrarily large number. In this way, q is orthogonal to every k vector except k_a and k_b, which mean we will have $k_i^T q = 0$ for $i \neq a, b$ and $k_i^T q = 1$ for $i = a, b$ resulting in attention weights equal to 0 expect for a and b. The large scalar value N provides higher values for $exp(k_i^T q)$ when $i = a$ or $i = b$, which leads to the attention weights a_a and a_b being closer to $\frac{1}{2}$ and the other attention weights closer to 0.

3.3 Question 3.c

3.3.1 Question 3.c.1

We can use q such that :

$$q = N * (\mu_a + \mu_b)$$

It will still work in the same fashion as in the previous question, as long as the covariance stays small (and thus the k vectors keep close to their respective μ means). However the higher α is for the covariance matrices the less it will work.

3.3.2 Question 3.c.2

For samples where the magnitude of k_a is small the resulting vector c will be mostly similar to v_b, while when the magnitude of k_a is big the resulting vector c will be mostly similar to v_a. This is simply because the result of $exp(k_a^T q)$ will be greatly dependent on the magnitude of k_a as it moves away from 1, leading to an exponential unbalance between the resulting attention weights a_a and a_b.

3.4 Question 3.d

3.4.1 Question 3.d.1

We can use q_1 and q_2 such that :

$$q_1 = N * \mu_a$$

$$q_2 = N * \mu_b$$

In this way, we will have $c_1 \approx v_a$ and $c_2 \approx v_b$ and consequently $c = \frac{1}{2}(c_1 + c_2) \approx \frac{1}{2}(v_a + v_b)$

3.4.2 Question 3.d.2

Using two attention heads with the previously defined q_1 and q_2, we should almost obtain $c \approx \frac{1}{2}(v_a + v_b)$ across samples. This is because the impact of the covariance of k_a will be way smaller, due to the attention head that uses the q_1 query vector being entirely "dedicated" to this item, whatever the magnitude of k_a.

4 Neural Machine Translation

4.1 Question 4.a

It is important to model the problem at the subword level because Cherokee is a polysynthetic language, which means it is a language mainly composed of morphemes that have meaning but cannot be used in isolation, and are thus used to form very long words with complex meaning. For example, in these languages a sentence can often be made with only one very long word. Thus, we need to interest ourselves in the subwords – the morphemes – as the variety of words that can be composed with these morphemes is too large to study.

4.2 Question 4.b

Intuitively, characters and subwords carry less meaning and are less varied than whole words. Thus, we do not need to encode as much information to create an embedding matrix that can represent characters and subwords correctly compared to whole words.

4.3 Question 4.c

The idea of Multilingual training is to learn multiple languages using a single model. Thus we hope than this singular model will generalize the task of learning a language, and be able to capture the representational similarity of these languages. The benefit of this generalization is a transfer of insight from one language to the other that allows for a better performance. The effect is existent but weak on high-resource languages (languages on which we have a lot of data), but becomes really effective

on low-resource languages (languages on which we have very few data). In other terms, this means that the model learns the general idea of a language through high-resource languages, and carries this information on to low-resource languages to learn them more efficiently despite their lack of data. This would typically be useful for the Cherokee language, on which the data is very scarce compared to French or English.

4.4 Question 4.d

4.4.1 Question 4.d.1

This error may be due to the model mixing up "hair" with "crown of daisies". Maybe these words are similar in the original language.
We would suggest augmenting the size of the dataset (that might be difficult for Cherokee...) as to have more example where "crown of daisies" appear, such that the model learns the difference.

4.4.2 Question 4.d.2

This error seems to be due to the model not understanding the grammar of the Cherokee language, as it did not manage to correctly identify the subject of the sentence (mixing "She" with "it").
To solve this problem, we would suggest a deeper model (for example, more hidden layers) to be able to understand the more complex concepts of grammar the Cherokee language might contain.

4.4.3 Question 4.d.3

This error probably means the model did not understand that "Littlefish" is a human name and just understood it at "small fish".
This error looks particularly hard to handle as few language work with such hard to identify proper nouns. Maybe a change in the model's architecture, a specific preprocessing, or simply a bigger model/dataset could help in understanding the difference.

4.5 Question 4.f

4.5.1 Question 4.f.1

According to the BLEU scores as seen on fig.5, the best translation is candidate 2 : "love can make anything possible". This seems correct considering the references, as the reference "love makes anything possible" is very close to the translation proposed by candidate 2.

```
compute_BLEU(candidate1, references, lambda_weights = [0.5, 0.5, 0, 0])
P0 : 0.6
P1 : 0.5
len(c) : 22
len(r) : 26
BP : 0.8337529180751805
0.45666528061553313

compute_BLEU(candidate2, references, lambda_weights = [0.5, 0.5, 0, 0])
P0 : 0.8
P1 : 0.5
len(c) : 31
len(r) : 29
BP : 1
0.6324555320336759
```

Figure 5: BLEU scores on candidate 1 and 2

4.5.2 Question 4.f.2

According to the new BLEU scores as seen on fig.6, the best translation is now candidate 1 : "the love can always do". Although with the full context of the problem we know this is not the best translation, it becomes the best one when considering the new references, as the now only reference "love can always find a way" is closer to candidate 1 than candidate 2.

```
compute_BLEU(candidate1, references, lambda_weights = [0.5, 0.5, 0, 0])
P0 : 0.6
P1 : 0.5
len(c) : 22
len(r) : 26
BP : 0.8337529180751805
0.45666528061553313

compute_BLEU(candidate2, references, lambda_weights = [0.5, 0.5, 0, 0])
P0 : 0.4
P1 : 0.25
len(c) : 31
len(r) : 26
BP : 1
0.316227766016838
```

Figure 6: BLEU scores on candidate 1 and 2, without reference 2

4.5.3 Question 4.f.3

As we have seen in the previous question, having only one point of reference for the translations might give an unfair advantage to medium quality translations compared to better translations that provide a different but valid turn of phrase. It could thus lead to a very "strict" metric, that punishes any translation that is not exactly the unique accepted answer even if the translation is valid.

4.5.4 Question 4.f.4

For the advantages, the BLEU score is a simple automated algorithm, so it is of course way faster and easier to score a big corpus of candidate translations compared to a manual evaluation by one or more human specialists. Another advantage is that the BLEU score does not need any knowledge of the source language or the translation process, it only operates on the result. On the other hand, using a human evaluator implies finding a specialist of both the source and target language.

For the disadvantages, the BLEU score is heavily dependent on the references translations as we have seen, which means that having too few or low quality references can lead to erroneous low or high BLEU scores. Also, the BLEU score is a method of evaluations that takes into account the precision of words only (and BP). This can result in a somewhat shallow metric, as it might not be able to correctly judge the importance of some errors in a sentence such as the words placement and order, that can completely alter the meaning.