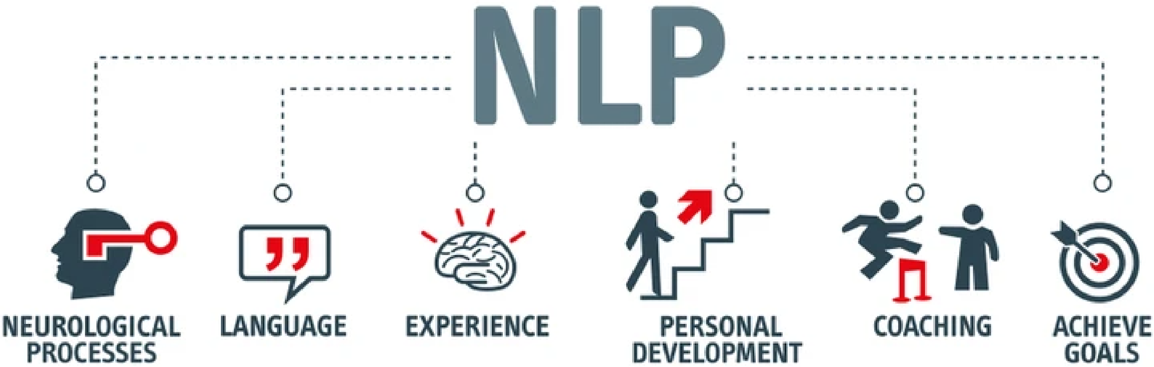


NLP3 Project

LAB02

January 2023

---

nelson.vicel-farah
antoine.zellmeyer
karen.kaspar
romain1.brand
maxence.plantard

Epita
SCIA - Promo 2023

Contents

1 Language Detection (24 points) – Guided coding

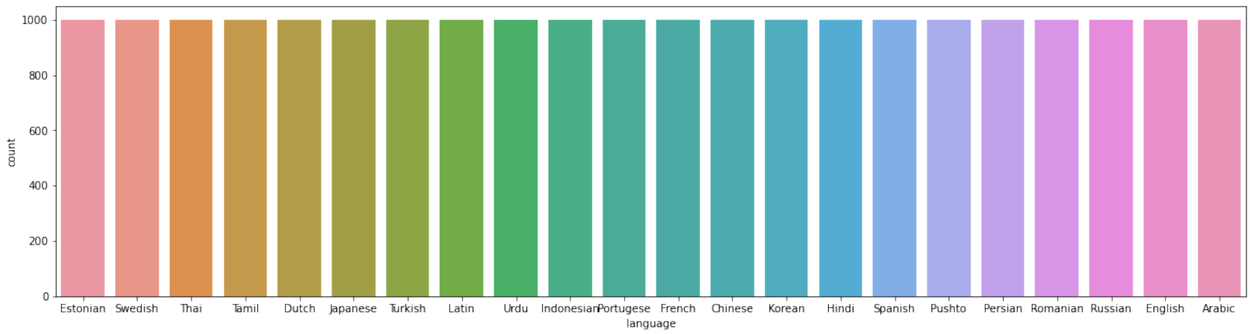
Question 0 (1 point)

Try out a translation of a French sentence in Google Translate (or Bing Translate) to English. What happens if you select the wrong source language as follows? Explain in a few sentences what is happening in the backend.

The backend of the translator will process the text as if it were written in the selected language. It will consider unrecognized word as if they were names or verbatim quotes, thus restoring them as is.

1.1 Question 1 (1 point)

Describe the distribution of languages and give at least two comments about the dataset. (1 point)



We count 22 languages that are equally distributed in the dataset. We can also note the sparsity of chosen languages for this dataset as they are really common languages (e.g English, Spanish) as much as less common and more vernacular languages (e.g Tamil, Urdu).

1.2 Question 2 (1 point)

Do the appropriate pre-processing to maximise the accuracy of language detection. What is your strategy?

Our pre-processing includes a few steps. The first step transforms uppercase characters into lowercase characters. We then remove punctuation. For specific languages that use logograms such as Japanse and Chinese, we have chosen to apply a specific pre-processing step that adds spaces between each character to allow the tokenization of words based on space-separated occurrence. That same treatment is applied for the Thai alphabet.

1.3 Question 3 (1 point)

What would be the problem if your dataset was unbalanced? (1 point)

The model would be highly biased toward languages that outnumber the other. We can also note that some languages would suffer from a lack of samples, which would make them harder to identify for a machine learning model that had been fed with these data.

1.4 Question 4 (1 point)

What techniques could you use to solve that?

- Oversampling: This involves increasing the number of samples in the minority class by duplicating existing samples or generating new synthetic samples.
- Undersampling: This involves reducing the number of samples in the majority class by removing some of the samples.
- Weighted loss function: We can use a weighted loss function to give more importance to the samples in the minority class during training.

1.5 Question 5 (4 point)

Train a model of your choice and describe the accuracy across languages. Use an 80%, 20% train-test split. Performance is not key but explain thoroughly the process and the metric(s) you are tracking.

We chose the Multinomial Naive Bayes algorithm to classify the texts. We get an accuracy of 98%.

	precision	recall	f1-score	support
Arabic	0.99	0.99	0.99	202
Chinese	0.99	0.98	0.99	204
Dutch	0.98	0.99	0.98	198
English	1.00	0.78	0.88	257
Estonian	0.96	1.00	0.98	192
French	0.99	0.96	0.98	208
Hindi	0.96	1.00	0.98	193
Indonesian	0.97	1.00	0.98	194
Japanese	0.99	1.00	0.99	198
Korean	0.97	1.00	0.98	194
Latin	0.93	0.99	0.96	188
Persian	0.99	1.00	0.99	198
Portugese	0.96	0.99	0.98	195
Pushto	0.96	0.99	0.98	193
Romanian	0.98	0.99	0.99	197
Russian	0.99	0.99	0.99	200
Spanish	0.99	0.99	0.99	199
Swedish	1.00	1.00	1.00	201
Tamil	0.98	1.00	0.99	196
Thai	0.99	1.00	1.00	199
Turkish	0.99	1.00	0.99	198
Urdu	0.98	1.00	0.99	196
accuracy			0.98	4400
macro avg	0.98	0.98	0.98	4400
weighted avg	0.98	0.98	0.98	4400

1.6 Question 6 (3 points)

Train a fasttext model on Tatoeba parallel corpus and check that performance is good.

The accuracy of fasttext is 0.958 on Tatoeba. The perfomance is therefore good.

1.7 Question 7 (3 points)

Test your fasttext model on the same dataset as in question 1-5. Compare with your custom model (make sure you use the exact same data for testing). How can you explain the difference in performance between the two models?

The accuracy of fasttext on the previous dataset is way lower than our Naive Bayes model. In this case, the accuracy of fasttext is **0.80** while the Naive Bayes accuracy is **0.98**.

This difference in performance can be explained by multiple factors :

- Both model have not been trained on the same distribution. Our custom model have been trained on the same dataset it has been tested on, while fasttext has only been trained on Tatoeba.
- FastText classifiers can struggle with very high-dimensional data because they rely on dense word vectors, which can become very large as the vocabulary size increases. Naive Bayes classifiers, on the other hand, can often handle high-dimensional data well because they make independence assumptions between features.
- FastText classifiers can also struggle with very sparse data, as the dense word vectors may not be able to capture the relationships between words in the data effectively. Naive Bayes classifiers, on the other hand, can often handle sparse data well because they make independence assumptions between features.

1.8 Question 8 (1 point)

Compute your performance metrics yourself and compare with sklearn.

The performance metric we rely on to compare models for this kind of task is the **accuracy** which is computed by.

$$\frac{\text{Right answers}}{\text{Wrong answers} + \text{Right answers}}$$

We eventually get the same results of the `accuracy_score` function from scikit-learn.

```
# compare with the sklearn metrics
print("Accuracy: ", accuracy(y_pred, y))

# sklearn
print("Accuracy: ", accuracy_score(y_pred, y))
################################################
```

```
Accuracy:  0.8009090909090909
Accuracy:  0.8009090909090909
```

1.9 Question 9 (2 points)

How could you improve the fasttext model performance from the previous question? Explain in a few sentences.

There are different ways to improve the fasttext model performance from the previous question:

- One way to improve the performance would be to train the fasttext model on the data it is actual tested on.
- We could also perform hyperparameter tuning. i.e we try different parameters and keep the ones that get the best performance.
- Applying preprocessing on the Tatoeba dataset.

1.10 Question 10 (1 point)

Which method would you use for language detection and why?

The most efficient method we tried seems to be the Multinomial Naive Bayes. While it depends on the context, MNB doesn't infer its responses from the assumption that words in sentences might be dependant. This means that the model focuses on more important language-specific cues : the alphabet and the lexicon. Another advantage of Naive Bayes is that it is way faster and simpler to train than fasttext.

1.11 Question 11 (2 points)

Given a sentence with N_1 tokens in English and N_2 token in French, what would be your strategy to assign a language to such sentence?

To determine the language of a sentence with N_1 tokens in English and N_2 tokens in French, we can follow the following steps:

- Identify all the words that are enclosed in quotation marks as they might indicate expressions or verbatim quotes.
- Exclude the quoted words from the count of N_1 and N_2
- We determine the language of the sentence by identifying the language with the most remaining tokens after the exclusion of the words between quotation. Therefore if N_1 has more tokens than N_2, we consider that the sentence is written in English. Alternatively, if N_2 has more tokens than N_1, we assume that the sentence is written in French.

1.12 Question 12 (4 points)

Would a multilingual architecture be robust to multiple languages in a single sentence? Elaborate your answer accordingly.

A multilingual model may be able to effectively handle multiple languages within a single sentence, depending on how it was trained and how it processes input.

There are several methods the model could use, such as identifying and processing each language separately or using a shared representation to process the entire sentence.

The model's ability to handle multiple languages in a single sentence will depend on its language identification and processing abilities, the complexity and structure of the sentence, and the diversity and representativeness of the data used to train the model.

2 Rotate two semantic spaces (23 points) – Not guided coding

2.1 Question 1 (1 point)

Explain in a few sentences how MUSE is doing the alignment of the semantic spaces in the supervised way.

MUSE is a tool that allows for the creation of a mapping from one language's word embeddings to another. In a supervised setting, MUSE uses a pre-existing dictionary that translates words between the two languages and the embeddings of both languages to learn the mapping. MUSE uses the Orthogonal Procrustes problem to find the optimal mapping, which involves finding an orthogonal matrix that can most accurately transform one set of embeddings into another. To do this, MUSE computes the matrix M by multiplying the matrices of the two sets of embeddings, and then uses Singular Value Decomposition to find the closest orthogonal matrix to M, which is equivalent to the desired mapping matrix.

2.2 Question 2 (2 points)

What is the limit of doing that alignment based on the approach taken in the supervised way?

There are several limitations to the supervised approach to aligning semantic spaces using MUSE:

- The approach requires a large amount of labeled parallel data, which consists of texts in both languages that are translations of one another. This can be difficult to obtain, especially for low-resource languages or specialized domains.
- The approach is limited by the quality of the parallel data. If the parallel data is noisy or contains errors, it can negatively impact the quality of the learned transformation matrix and the alignment of the semantic spaces.
- The approach is sensitive to the choice of supervised learning algorithm and hyperparameters. Different algorithms and hyperparameter choices can lead to different results, and it may be difficult to determine the optimal settings without a large amount of trial and error.
- Finally, the approach is limited to aligning the semantic spaces of two languages. If more than two languages are involved, additional transformation matrices would need to be learned and combined, which can be a complex and time-consuming process.

2.3 Question 3 (2 points)

How can we align two semantic spaces in a domain specific field, e.g., in a tech company?

To perform this task, we can follow the following steps:

- Collect a large amount of parallel data that is relevant to the specific domain (in this case the tech company). The data must consist of texts in both languages that are translations of one another.
- Preprocess the data to remove noise and errors, such as typos and non-standard language.
- Create word embeddings for each language using an unsupervised approach, such as Skip-Gram or GloVe.
- Use a supervised learning algorithm, such as Procrustes analysis or Orthogonal Procrustes, to learn a transformation matrix that maps the embeddings of one language to the other using the parallel data.
- Use the transformation matrix to align the semantic spaces of the two languages by applying it to the word embeddings of one language.

It may also be helpful to use domain-specific data or techniques to fine-tune the alignment for the tech domain. This could include using additional parallel data that is specific to the tech domain or incorporating domain-specific knowledge into the transformation matrix.

2.4 Question 4 (5 points)

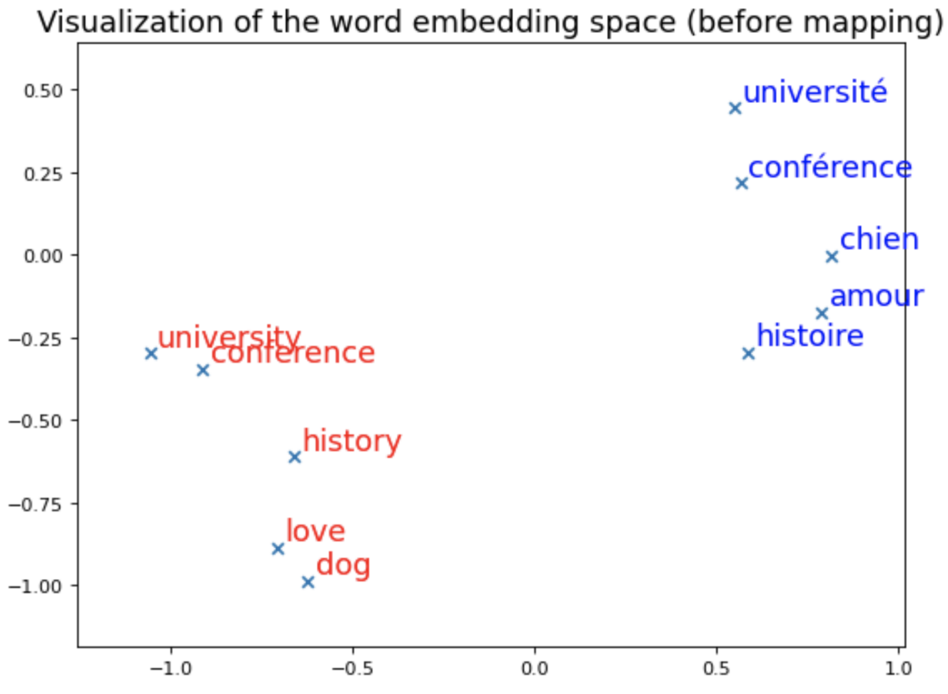
Align the French space and the English space together, with the method of your choice.

Firstly, we load the pre-trained monolingual word embeddings from *fastText Wikipedia embeddings*. Then, to align French and English embeddings, we use the unsupervised method of MUSE. This gives us a linear mapping of 300x300 weights that we can use to align the French embeddings.

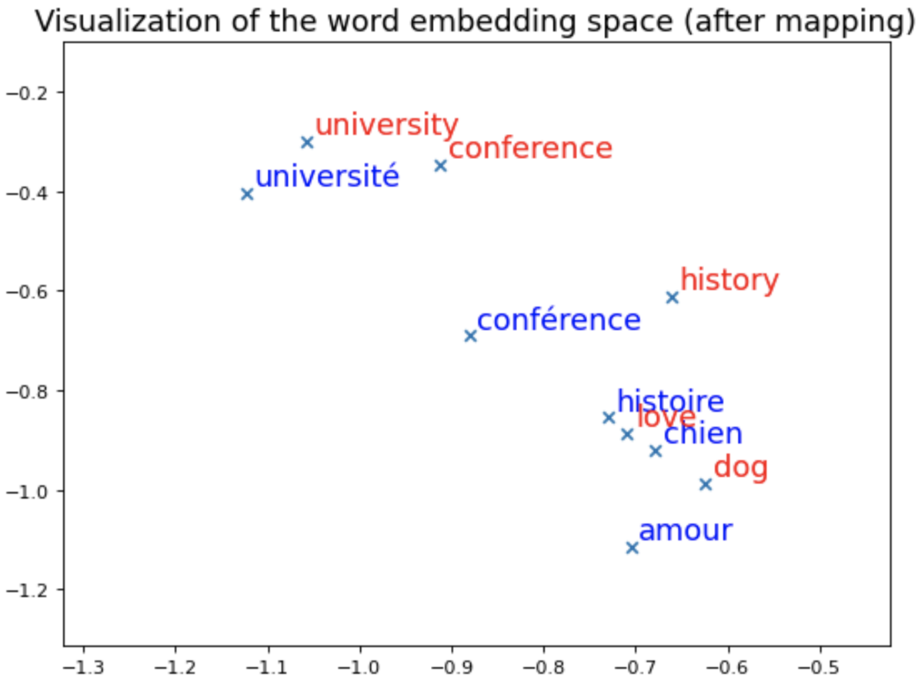
2.5 Question 5 (2 points)

Visualize the output on a few words of your choice. Comment on the performance of the alignment based on the output.

To render a good visualization of the 300-dimensional embeddings we use the Principal Component Analysis to project the data on 2 dimensions. Despite the obvious information loss, we obtain a decent representation of the semantic space :



Before applying the mapping, there is no alignment with relation to the meaning whatsoever. It lumps together the words from the same language, creating two groups of words, one in each language, with nothing indicating that one word is a translation of the other in French.



After applying the mapping, we notice a considerable increase in performance. The words from a language aren't lumped together anymore and most translations are close to each other. We also notice that words that have a common context (e.g university and conference) are closer to each other than words with unrelated meaning.

2.6 Question 6 (2 points)

How can you find the translation of a word with this approach? Explain your method and the distance metric you choose in a few sentences.

To find a translation of a word from aligned embeddings, the process is to look for the nearest neighbor of the embedding on the other language semantic space. To make this operation succeed, we need a metric to compute the distance between two data points. For this step we chose the **cosine similarity**. We could also have used the **inner product** if we had chosen to normalize the embeddings beforehand.

2.7 Question 7 (3 points)

Apply your approach and comment on the performance of the translation.

```
Nearest neighbors of "université":
0.8894 - university
0.7790 - faculty
0.7130 - universityuniversity
0.6767 - professor
0.6703 - wisconsin–madison

Nearest neighbors of "amour":
0.7338 - love
0.6439 - unrequited
0.6349 - longing
0.6126 - lover
0.6063 - affection

Nearest neighbors of "histoire":
0.7363 - history
0.5442 - prehistory
0.5373 - histories
0.5295 - historiography
0.5231 - historical

Nearest neighbors of "conférence":
0.7318 - conference
0.6318 - conferences
0.5806 - meeting
0.5565 - plenary
0.5462 - convened

Nearest neighbors of "chien":
0.7598 - dog
0.6263 - dogs
0.6057 - poodle
0.5823 - puppies
0.5797 - puppy
```

We observe that the translations are rather accurate although some irrelevant results may appear in the nearest neighbors of some word (e.g "wisconsin-madison" as a translation of *université*).

2.8 Question 8 (4 points)

What is the limit of aligning two spaces at a sentence level? What do you suggest to improve the alignment, at a sentence level?

When aligning two spaces at the sentence level, there are a few potential challenges that may arise:

- Sentence structure: Sentences can have different structures and may use different syntactic constructs, which can make aligning the spaces more difficult.
- Semantic meaning: Sentences can convey different meanings, even if they use

the same words. This can make it hard to align the spaces, as the words in the sentences may be in different contexts and have different connotations.

- Language differences: If the two spaces are in different languages, aligning them at the sentence level may be more challenging due to differences in grammar, syntax, and idiomatic expressions.

To improve alignment at the sentence level, one approach is to use supervised methods that rely on labeled data. For example, we can use parallel data (sentences in two languages that are translations of each other) to train a model to align the spaces. Additionally, we can use cross-lingual transfer learning, where we pre-train a model on a large corpus of one language and then fine-tune it on a smaller dataset in another language. These methods can help the model learn the structural and semantic similarities and differences between the two languages.

Other techniques that can be helpful to improve alignment at the sentence level include the use of bilingual dictionaries, machine translation techniques, or use of multilingual models with alignments within their sub-spaces.

2.9 Question 9 (2 points)

Someone, in your company, asked you to do sentiment analysis on their dataset. Given a set of sentences $s_1, \dots, s_N$, where s_i can be written in any language, what architecture would you use to have a vector representation of s_i? Motivate in 2-3 bullet points.

When it comes to sentiment analysis, there are several architectures than can be used to create a vector representation of a sentence:

- Utilizing pre-trained word embeddings like word2vec or GloVe to create a vector representation of each word in the sentence. They are trained on large sets of text data to get both the semantic and syntactic properties of the words. In order to generate a sentence representation, we can either average or sum the word vectors or even apply more advanced techniques such as weighting the word vectors based on their significance within the sentence.

- We can also use Recurrent Neural Networks (RNNs) like LSTM to generate a vector representation of the sentence. RNNs are particularly effective in processing sequential data. They are able to remember information from previous steps, thus allowing them to grasp the context and meaning of words within a sentence.

- Transformer-based models such as BERT, RoBERTa and GPT-2 are the state-of-the-art architectures which can be used to encode sentences and generate a contextualized vector representation of them. These models are pre-trained on vast amounts of text data, produce high-quality embeddings and can be fine-tuned for specific tasks like sentiment analysis.

2.10 Question 10 (5 points)

How would you do sentiment analysis across multiple languages in a domain specific context? Justify your approach step by step.

Doing sentiment analysis across multiple languages in a domain-specific context can be challenging, as it requires both a good understanding of the domain-specific language and the ability to perform sentiment analysis in multiple languages. Here's an approach one could take to perform this task:

- Data collection: We start by collecting a large dataset of domain-specific text in multiple languages. This dataset should include both positive and negative examples of sentiment in the domain of interest.
- Preprocessing: Perform text preprocessing on the collected data, such as lower-casing, removing punctuation, tokenizing or removing stop words.
- Language Detection: One should use language detection techniques to identify the language of the text like we did in the previous part. It could be done by using popular libraries such as `langdetect` or our Naive Bayes model.
- Aligning Embeddings: After language detection, we embed the words in the same semantic space using techniques like MUSE. We should for this step align the languages to the english space as a lot of pre-trained model exist for english sentiment analysis. We could also consider to normalize the embeddings which would allow faster inference and better alignment with language too far apart from english (e.g Chinese).
- Sentiment Analysis: Next, we apply sentiment analysis techniques on the aligned data. There are several pre-trained models that we can use, such as BERT, RoBERTa, or ALBERT. Fine-tuning these models on the domain-specific dataset can also give good results.
- Domain-specific fine-tuning: We should then consider fine-tuning the model on a domain-specific dataset to make sure that the model is well-suited to the domain.
- Evaluation: Finally, evaluate the model's performance on a test dataset to see how well it is able to detect sentiment in the domain-specific text.

3 Attention Exploration (22 points)

3.1 Question a (2 points)

Describe (in one sentence) what properties of the inputs to the attention operation would result in the output c being approximately equal to v_j for some $j \in \{1, ..., n\}$. Specifically, what must be true about the query q, the values $\{v_1, ..., v_n\}$ and/or the keys $\{k_1, ..., k_n\}$?

$$\text{with } c = \sum_{i=1}^{n} a_i v_i$$

$$\text{and } a_i = \frac{\exp(k_i^T q)}{\sum_{j=1}^{n} \exp(k_j^T q)}$$

c tends to v_j when a_j tends to 1 while every other a_i tends to 0, which happens when $k_j^T q$ is substantially higher than every other $k_i^T q$.

3.2 Question b (4 points)

Give an expression for a query vector q such that the output c is approximately equal to the average of v_a and v_b, that is, $\frac{1}{2}(v_a + v_b)$.

To achieve this, we need a_a and a_b to tend to $\frac{1}{2}$ while every other a_i tend to 0. This occurs when

$$q = \lim_{C \to \infty} C * (k_a + k_b)$$

In a numerical setting, C can be an arbitrary large scalar. In this case, q is neither perpendicular to k_a nor k_b but stays perpendicular to every other vector k_i. Which gives us :

$$a_a = \frac{\exp(C * k_a^T(k_a + k_b))}{\exp(C * k_a^T(k_a + k_b)) + \exp(C * k_b^T(k_a + k_b)) + \sum_{i \notin \{a,b\}} \exp(C * k_i^T(k_a + k_b))}$$

$$= \frac{\exp(C * k_a^T(k_a + k_b))}{\exp(C * k_a^T(k_a + k_b)) + \exp(C * k_b^T(k_a + k_b)) + (n-2)}$$

$$= \frac{\exp(C * (k_a^T k_a + k_a^T k_b)))}{\exp(C * (k_a^T k_a + k_a^T k_b))) + \exp(C * (k_b^T k_a + k_b^T k_b))) + (n-2)}$$

$$= \frac{\exp(C * ||k_a||^2 + 0))}{\exp(C * ||k_a||^2 + 0)) + \exp(0 + C * ||k_b||^2)) + (n-2)}$$

$$= \frac{exp(C)}{2exp(C) + n - 2} \text{ (the norm of key vectors is 1)}$$

The same goes for a_b. Every other attention weight a_i for $i \notin \{a, b\}$ can be described as follow :

$$a_i = \frac{1}{2exp(C) + n - 2}$$

which means that C has to largely outweigh $\log(n-2)$ in order to make $n-2$ insignificant and allow a_a and a_b to approximate $\frac{1}{2}$.

3.3 Question c (5 points)

3.3.1 i (2 points)

Design a query q in terms of the μ_i such that as before, $c \approx \frac{1}{2}(v_a + v_b)$, and provide a brief argument as to why it works.

We can adapt the query from the previous question for this case, which gives the following query :

$$q = \lim_{C \to \infty} C * (\mu_a + \mu_b)$$

This happens due to a being vanishingly small in the covariance matrix $\Sigma_i = aI$ which implies that $k_i \approx \mu_i$.

3.3.2 ii (3 points)

When you sample $\{k_1, ..., k_n\}$ multiple times, and use the q vector that you defined in part i., what qualitatively do you expect the vector c will look

like for different samples?

For a vanishingly small a we can assume that $k_a \sim \mathcal{N}(1, \frac{1}{2})\mu_a$. If we use the results from **question b** for a large C that makes $(n-2)$ insignificant we get:

$$\begin{aligned}
&\text{Let X be defined by } X \sim \mathcal{N}(1, \frac{1}{2}) \\
c &\approx \frac{exp(C * X)}{exp(C * X) + exp(C)} v_a + \frac{exp(C)}{exp(C * X) + exp(C)} v_b \\
&\approx \frac{1}{exp((1 - X) * C) + 1} v_a + \frac{1}{exp((X - 1) * C) + 1} v_b
\end{aligned}$$

We can conclude that c fluctuates between v_a and v_b as we sample the key vectors multiple times.

3.4 Question d (3 points)

3.4.1 i (1 points)

Design q_1 and q_2 such that c is approximately equal to $\frac{1}{2}(v_a + v_b)$

We assume that the covariance matrices are $\Sigma_i = \alpha I$.
We know that the final output of the multi-headed attention is the average of each head: $c = \frac{1}{2}(c_1 + c_2)$, therefore we need to have $c_1 \approx v_a$ and $c_2 \approx v_b$. This can be achieved with the following queries :

$$q_1 = C * \mu_a \text{ and } q_2 = C * \mu_b$$

. We hence have:

$$\begin{aligned}
c_1 &= \sum_{i=1}^{n} \alpha_i * v_i \\
&\text{if i is different from a we have:} \\
\alpha_i &\approx \frac{exp(0)}{exp(C)} \approx \frac{1}{exp(C)} \approx 0 \\
&\text{otherwise :} \\
\alpha_a &\approx \frac{exp(C)}{exp(C)} \approx 1
\end{aligned}$$

Therefore all the elements of the sum are cancelled except for the element at the index a: hence $c_1 \approx v_a$
In the same manner $c_2 \approx v_b$
So $c \approx \frac{1}{2}(v_a + v_b)$

3.4.2 ii (2 points)

Take the query vectors q_1 and q_2 that you designed in part i. What, qualitatively, do you expect the output c to look like across different samples of the key vectors? Please briefly explain why. You can ignore cases in which $q_i^T k_a < 0$.

For a vanishingly small a we can write that $k_a \sim \mathcal{N}(1, \frac{1}{2})\mu_a$ while for any other i, $k_i = \mu_i$.

Let X be defined by $X \sim \mathcal{N}(1, \frac{1}{2})$
The query vectors that we designed before are:
$q_1 = C * \mu_a$ and $q_2 = C * \mu_b$
We have seen that the value is cancelled for all indexes except when i = a.
For $i = a$, we have:

$$
\begin{aligned}
c_1 &\approx \frac{exp(k_a^T q_1)}{exp(k_a^T q_1)} v_a \\
c_1 &\approx \frac{exp(X * \mu_a^T * C * \mu_a)}{exp(X * \mu_a^T * C * \mu_a)} \approx \frac{exp(X * C)}{exp(X * C)} v_a \\
&\approx v_a \text{ because C outweigh X and X} > 0
\end{aligned}
$$

Therefore $c_1 \approx v_a$ and in the same manner $c_2 \approx v_b$

For $i = b$, and c_2 we have

$$
c_2 \approx \frac{exp(k_b^T q_2)}{exp(k_b^T q_2) + exp(k_a^T q_2)} v_b + \frac{exp(k_a^T q_2)}{exp(k_b^T q_2) + exp(k_a^T q_2)} v_a
$$

$$
c_2 \approx \frac{exp(\mu_b^T * C * \mu_b)}{exp(\mu_b^T * C * \mu_b)} \approx \frac{exp(C)}{exp(C)} v_b \approx v_b
$$

We end up with the same result as in the first question with $c \approx \frac{1}{2}(v_a + v_b)$ because the magnitude of k_a does not influence the result enough.