EXPLORATORY VISUAL ANALYSIS OF MACHINE TRANSLATION SYSTEMS

Elora Lacroux

Supervisors :
Marta Ruiz Costa-Jussà
Pere Pau Vázquez Alcocer

4 July 2019

ABSTRACT

Understanding how Machine Learning systems work exactly has become a real challenge for domain experts. Using visualisation tools is an interesting approach for providing an overview of complex models.

Data Visualisation is nowadays an expanding field, especially for Artificial Intelligence, since the intelligent systems become more complex and accurate over time.

During this project, we focused on a specific **Machine Translation** system based on an **Interlingua** model.

The goal of this project is to provide visualisation tools for helping researchers to understand their model. This way, we created 2 different tools which cover 3 different use cases. By using **UMAP**, a **dimension reduction** technique, we are now able to visualise multilingual **intermediate representations** at the level of sentences and words. We can also visualise multilingual representations from **decoders layers** and finally, we can identify gender bias in Contextual Words Embeddings.

The tools have been implemented by using **Python** and **Bokeh**, a data visualisation library.

Keywords: Data Visualisation, Machine Translation, Interlingua, UMAP, Dimension Reduction, Intermediate representations, Decoders Layers, Python, Bokeh

ACKNOWLEDGMENT

First, I would like to thank my two supervisors Marta Ruiz Costa-Jussà and Pere Pau Vázquez Alcocer for their support and guidance during this project. Their insights had been very helpful.

Then, I would like to thank Carlos Escolano Peinado for providing me all the datasets. His explanations were a valuable help.

And finally, I would like to thank Jean Blonblon for his constant support and for knowing how to read between the lines.

CONTENTS

INTRODUCTION

With all the fast progress in Artificial Intelligence and especially Machine Learning (ML), Data Visualisation has become a serious concern for ML researchers. Indeed, in ML, the systems learn by themselves and de facto it is rarely simple to understand how they work. Even if the results are good, experts can't explain exactly how the system computes yet. This way, visualising the results of the intermediate steps and thus the journey of the data in the system is one of the easiest way to lift the veil on the AI black box. Our main goal is then to provide tools for understanding how this kind of systems works.

During this project, we focused on a concrete Machine Learning system : a Neural Machine Translation system. This system is multilingual and aims to provide a universal intermediate representation of languages called Interlingua. To make sure the Interlingua is properly created, we decided to create a tool to visualise directly the intermediate representations of each language from the model. By reducing the dimension of the vectors from 512 to 2, we are now able to display them on scatter plots. This tool has 2 levels of granularity since it allows to visualise the sentences and the words.

The studied model is based on combining variational autoencoders with encoder-decoders [1]. Thus, in order to obtain a better understanding of this system, we also provide a tool for analyzing the multilingual representations from the decoders layers. This tool highlights the journey of the sentences through the translation process. The parallel sentences can be compared and the most distant are identified.

However, the application is not limited to Machine Translation and multilingual systems. With this tool, we can also identify gender bias from contextual words embeddings.

This report will be divided in 4 main parts. First, I will provide some explanations on Neural Networks in the field of Machine Translation, the principles of dimensionality reduction and the current uses of Data Visualisation in AI. Then, I will present the app and each part of the visualisation tools. Moreover, I will explain more in depth the use cases of the tools. And finally, I will provide the future of the app and the others possible use cases.

With the increase of available data, Machine Learning algorithms have become more and more used and efficient. Machine Learning covers a lot of different problems such as classification, regression, clustering, dimensionality reduction or pattern recognition. Different types of learning exist and the models become very complex, especially Neural Networks architecture. Since this kind of model learns by themselves, in most cases, it is quite difficult for the researchers to properly understand how the different stages of the system affect the overall learning process. The goal of this project is to help the experts to get a better understanding on how a concrete type of Machine Learning systems work : Neural Machine Translation systems, by using visualisation tools.

1. Neural networks in Machine Translation

The aim of this project is mainly to help researchers to visualise and to improve their machine translation models. The main model studied during this project is introduced in the paper "(Self-Attentive) Autoencoder-based Universal Language Representation for Machine Translation" [1]. Before giving an overview of this model and its architecture, I will introduce some useful notions to understand more in depth the concerns behind the visualisation tools.

a. Encoder-Decoder structure

The first Neural Machine Translation (NMT) systems were based on this structure. Basically, the structure is divided in 2 parts : the Encoder and the Decoder.

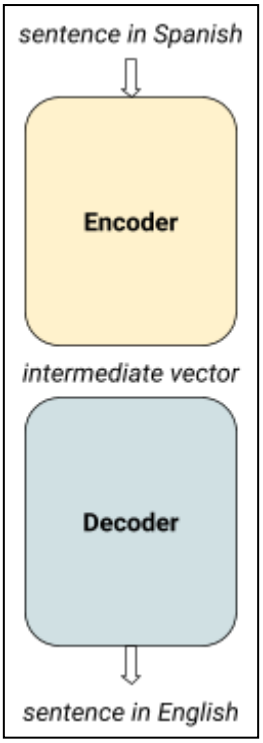


Figure 1 - Encoder-Decoder architecture

The goal of the Encoder is to transform a sentence into a high dimensional vector understandable for the Decoder, and the goal of the Decoder is to, from this vector, obtain the most suitable translation.

Each part can manage one language. For instance, if we want to translate Spanish to English and English to Spanish, 2 pairs of Encoder-Decoder are needed. In this kind of architecture, the intermediate vector isn't the same in both direction.

b. Autoencoder structure

The aim of autoencoders is to be able to reproduce the inputs in the outputs. Basically, the work of autoencoders is to, given an input, compress this input into an intermediate representation and then, reconstruct the input in output from this minimized representation.

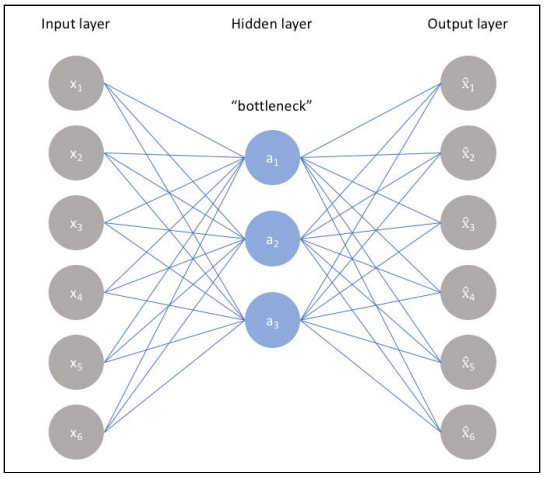


Figure 2 - Autoencoder architecture [2]

The autoencoder architecture is composed of 2 parts as the Encoder-Decoder structure : one for the encoding and one for the decoding.

- Encoding : The encoding part can compress the input into a lower dimensionality space.
- Decoding : The decoding part reconstructs the input from the intermediate representation in the low-dimension space.

c. Overview of the studied architecture

In [1], a new architecture for Machine Translation is proposed. The main goal of this architecture is to provide a shared intermediate representation between all the languages. This architecture is based on autoencoders. This structure allows to train both directions in translation and both decoders at the same time.

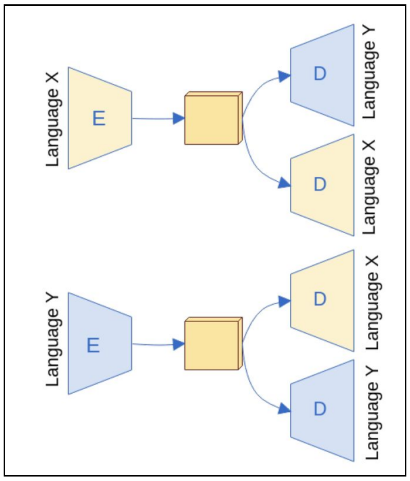


Figure 3 - Architecture example [1]

As shown in the Figure 3, the goal is that each encoder compresses the input no matter the language to a unique intermediate representation. Then, from this intermediate representation, the decoders of each language are trained. Since the model can translate from one language to another through this intermediate representation, it is highly likely that this representation is common to all languages. Such a representation is commonly called Interlingua [3]. Obtaining a universal language representation is nowadays one of the main goal in Machine Translation.

Initially, the model has been trained for Spanish-English translations. Training this model consists in minimizing these terms :

- the wrong words produced by the decoders given the representations for the 4 directions :
  - English to Spanish,
  - English to English,
  - Spanish to English,
  - Spanish to Spanish.
- the distance between the intermediate representation from each encoder on the same sentences, since this representation is supposed to be universal.

One of the advantages of this system is that a new language can be added without re-training the other languages. Currently, the model is trilingual : it can understand and translate English, Spanish and French sentences.

The French language was added by using only French-English sentences, it is an important point for later in the part 2.1.2.ii (Decoder preprocessing). To add the French language, it was enough to train its encoder with the English decoder already trained. Furthermore, as the Spanish and English decoder understand each other, the Spanish decoder is able to translate the French representation without being trained directly to do this task.

2. Dimensionality reduction

The intermediate representations of our system and the output of the decoder layers are 512-dimensional vectors. Thus, they can't be analyze directly by using classical tools or visualisation methods. As a consequence, the first task was to reduce the dimensionality, as in many high-dimensional visualisation methods.

a. Overview

Dimension reduction algorithms are often used in the field of visualisation. In Machine Learning, researchers generally manipulate datasets with hundreds of features. The aim of these algorithms is to reduce the dimension without losing information as much as possible. This kind of algorithms wouldn't be useful if the reduced dimensions didn't provide information on the high-dimensional set.

Many algorithms exist, and each of them provides a different kind of projection. There are basically 2 types of dimension reduction methods :

- *Linear methods* :

These methods are the most common and consist in applying a linear transformation on the data. For instance, the Principal Component Analysis (PCA) method projects the data along the principal component axis.

- *Non-linear methods* :

These methods, also called manifold learning methods, are used for high-dimensional data. In the scope of this project, this kind of methods is needed, because the data aren't isomorphic to a linear subspace. The autoencoders seen before can be used for this task. However, the technique we used is the Uniform Manifold Approximation and Project (UMAP). Another alternative would have been to use the t-SNE algorithm, it is more famous than UMAP, but UMAP is faster in terms of computing time [4].

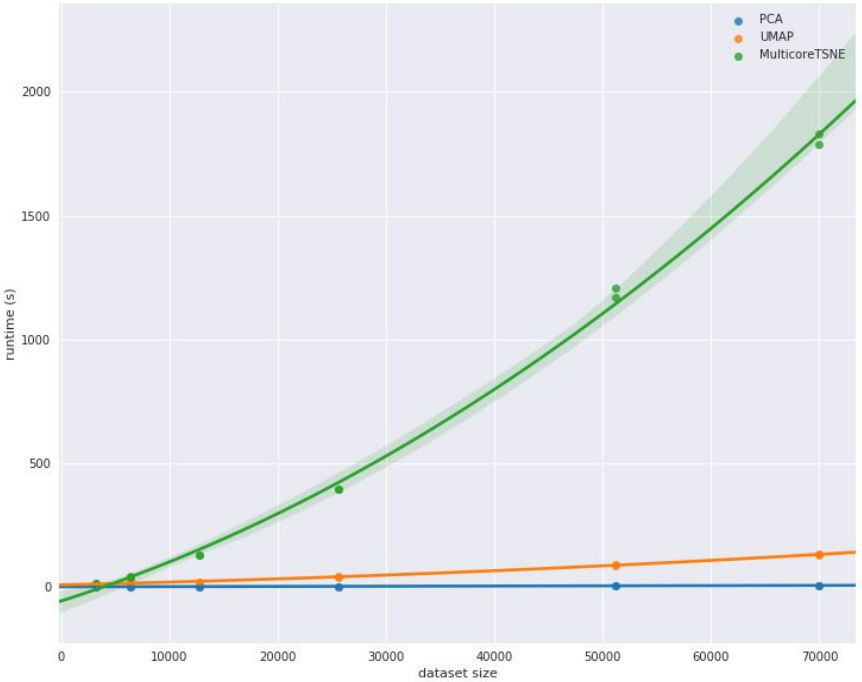


Figure 4 - Runtime performance of PCA, UMAP and a fast t-SNE version [5]

As shown in Figure 4, UMAP is clearly faster than t-SNE. PCA is the fastest, however, it can't be used since our data aren't linear.

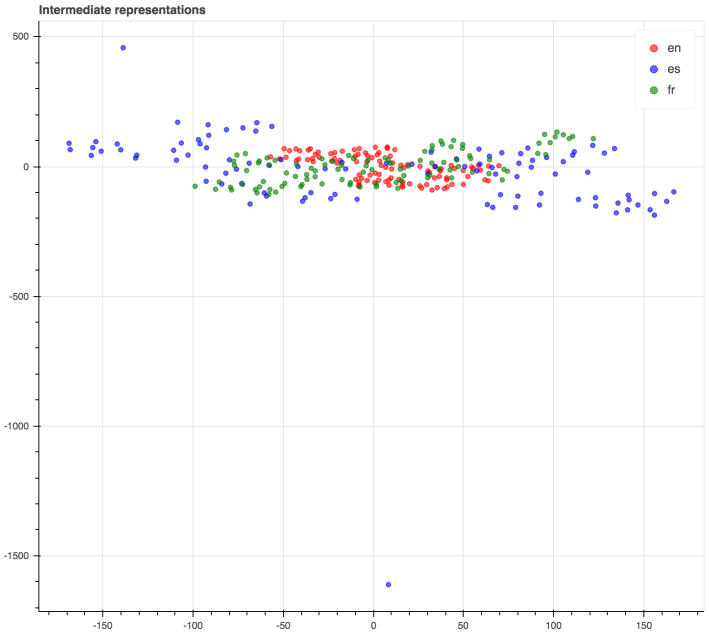


Figure 5 - Applying t-SNE on our sentence dataset

b. Uniform Manifold Approximation and Projection (UMAP)

This nonlinear method seeks to learn the structure of the data and thus, to give a low dimensional embedding without losing the topological structure of the initial data. This projection won't be unique, as shown in the Figure 6, even if we have the same parameter at the beginning, this projection will keep the same properties.

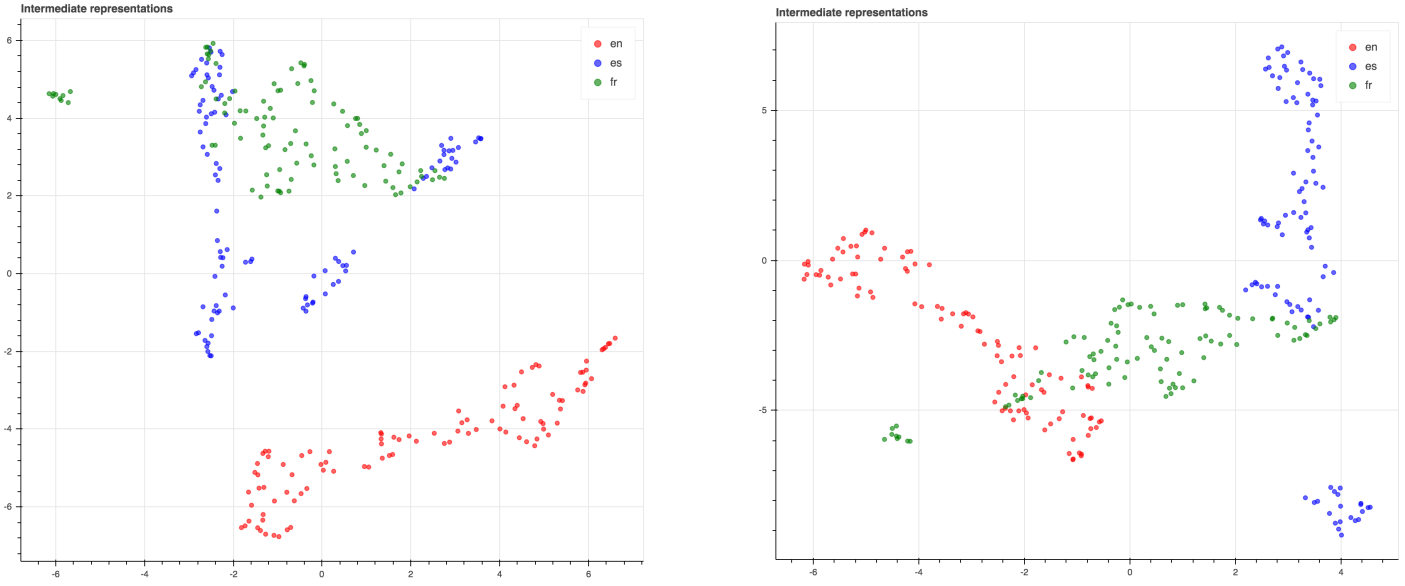


Figure 6 - Applying UMAP on same sentence embeddings

There are 4 main UMAP parameters :

- `n_neighbors` : This parameter indicates how the algorithm will consider the neighbourhood of each element. With a low value, the algorithm focuses on local structure, while with a large value, the algorithm focuses on global structure since it considers a larger neighbourhood. We set this value to 10, thus we don't lose too much of the details.

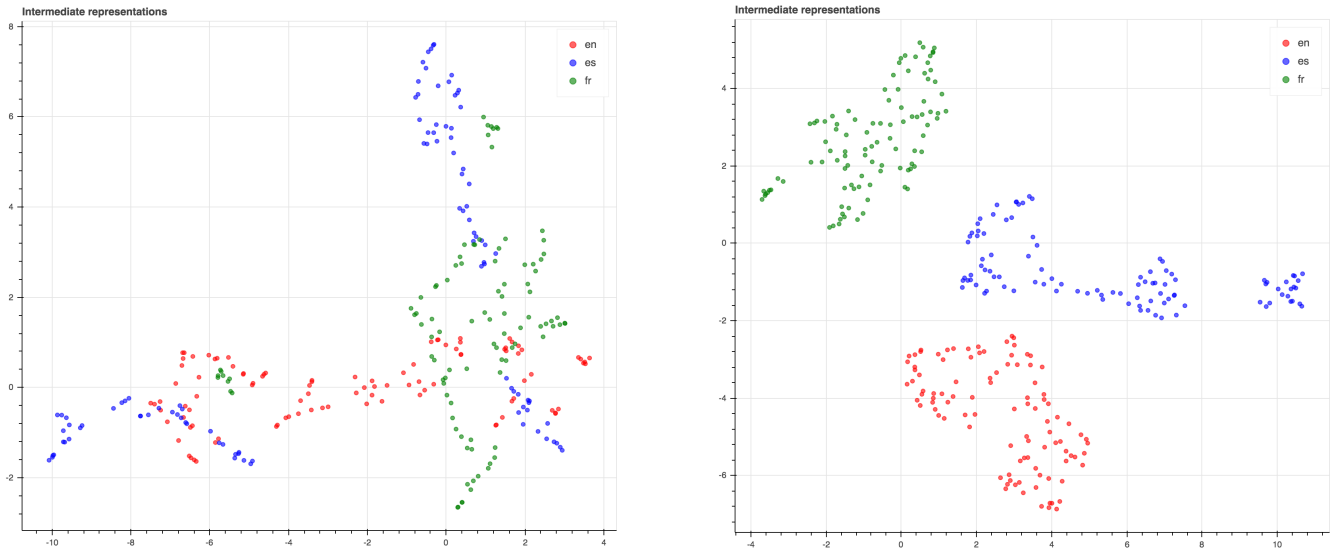


Figure 7 - Changing UMAP n_neighbors parameter :
left : `n_neigbors = 5`
right : `n_neighbors = 20`

- `min_dist` : This parameter provides the minimum distance between 2 points to being packed together. We set this value to 0.005.

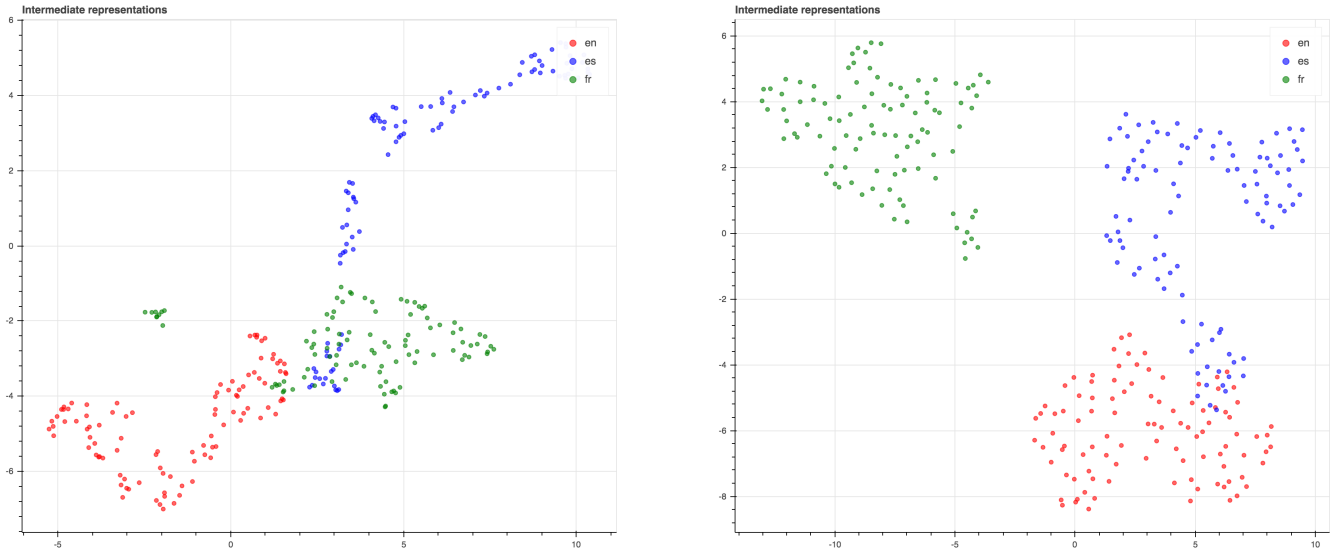


Figure 8 - Changing UMAP min_dist parameter :
left : `min_dist = 0.01`
right : `min_dist = 0.5`

- `n_components` : This parameter allows the user to set the dimension of the output data. Our output are 2-dimensional.
- `metric` : This parameter sets the metric to use in order to compute the distances between the points. We used the correlation metric.

3. Data Visualisation in AI

a. Applications

Data Visualisation is a very important step, especially for Artificial Intelligence. Since the systems learn by themselves, it's not always so easy to understand how an intelligent system makes a choice. The AI black box is still a hot topic. Providing and using visualisation tools that humans can understand helps researchers to have an overview and a deeper understanding of their models.

Data Visualisation tools can be used in different ways :

- For explanation : Data visualisation explanation tools are used to explain the data to someone else. For instance, an infographic aims to explain data to others.

Visual explanations for AI become more and more important with the increasing complexity of intelligent systems. On this subject, the workshop VISxAI on Visualization for AI Explainability [6] promotes visual explanations on different AI topics such as Dimensionality Reduction or Neural Networks.

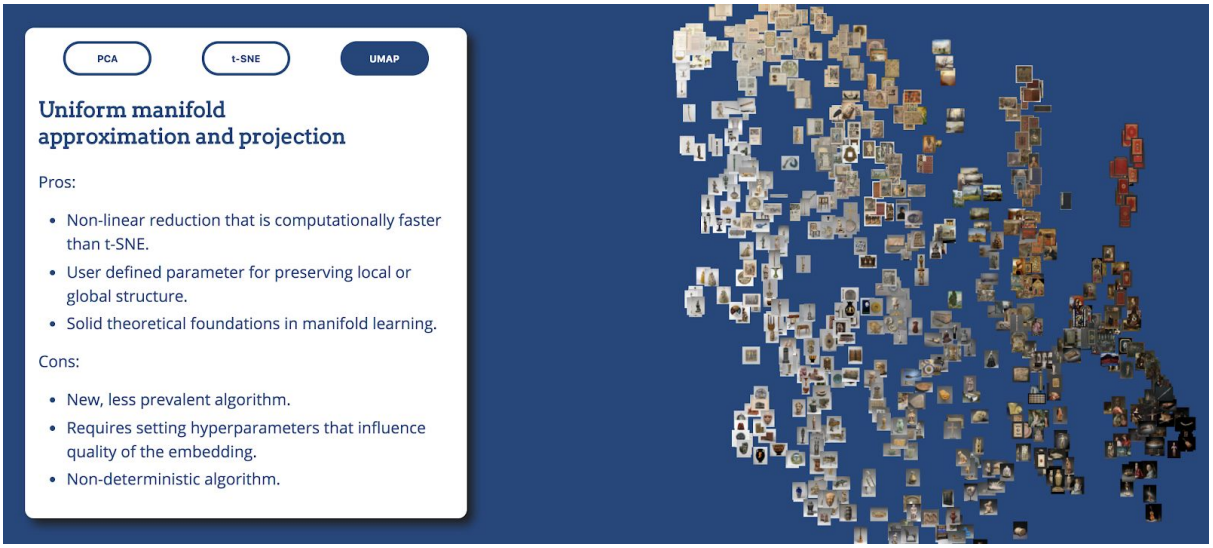


Figure 9 -Visual explanation of Dimensionality Reduction from VISxAI [7]

- For exploration : This kind of data visualisation is useful to obtain a better understanding of data that users don't know well. It is interesting in an Exploratory Data Analysis step. The visualisation tool created during this project could be considered as an exploration tool.

One of the most famous tools for AI researchers is the TensorBoard to visualise TensorFlow graphs and to make easier the debugging and the understanding of TensorFlow Neural Networks.

The Neural Network playground for TensorFlow is also a simple tool to visualise and to easily test different types of network architecture.

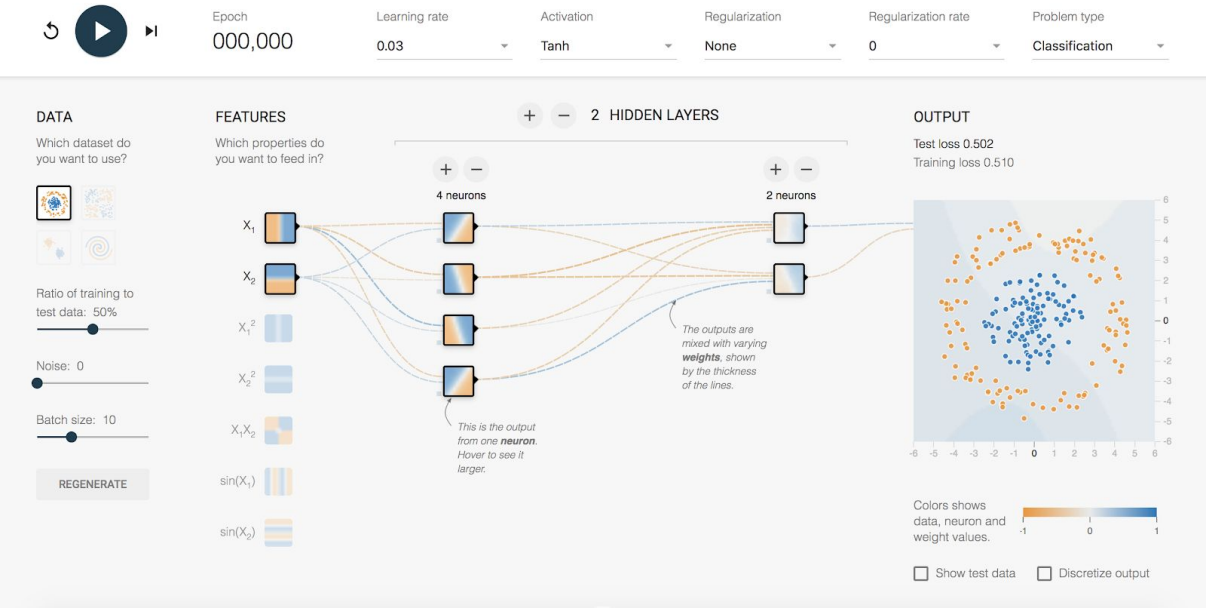


Figure 10 - TensorFlow Playground [8]

b. Related work

As seen before, some tools can be used in AI to explore and understand the data. In Natural Language Processing and especially in Machine Translation, a few tools exist.

The research group Google Big Picture has released in 2016 its open-source project Embedding Projector [9] which allows to explore word embeddings and apply different dimension reduction techniques.

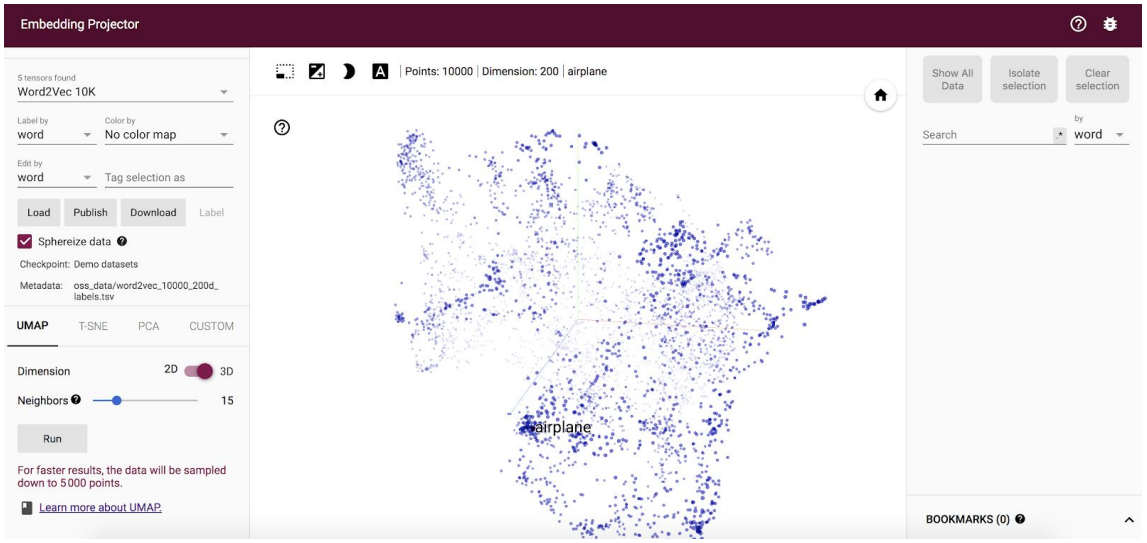


Figure 11 - Embedding Projector [10]

In the field of Machine Translation, such a tool doesn't exist in open-source for multilingual model visualisation. However, in the paper "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation" [11], the researchers offer a visual analysis for their multilingual model and visual evidences for an interlingua representation.

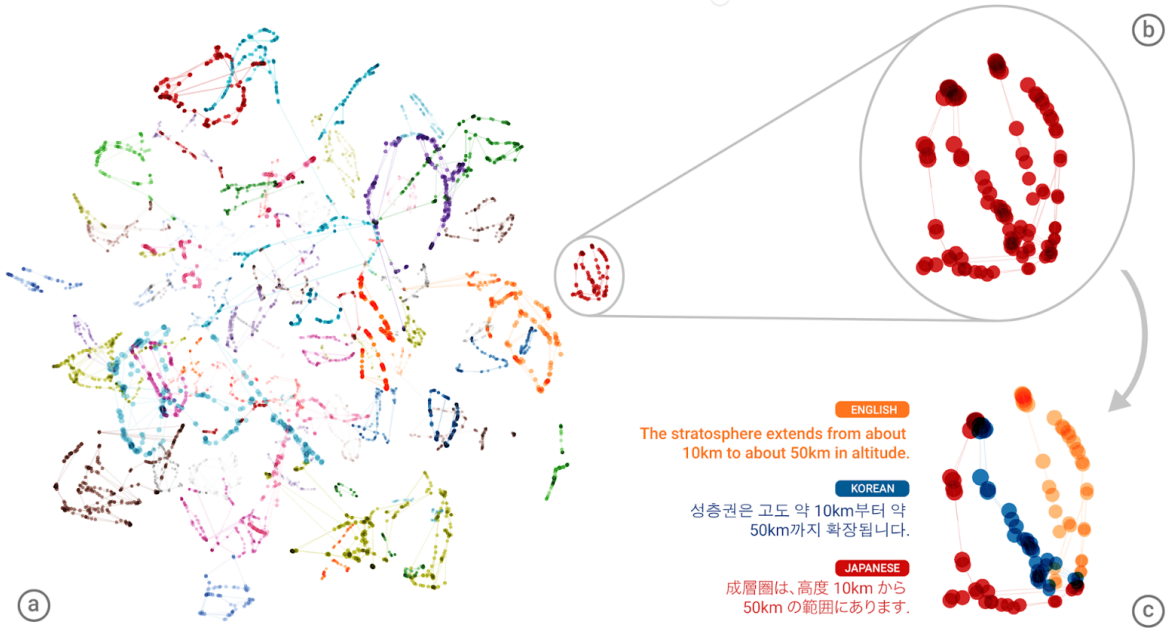


Figure 12 - Visual representation from Google's paper [11]

In Figure 12, each color represents a different trained model and each point a step in the decoding process. They compared 74 sentences translated in the 3 languages. This kind of visualisation gives a visual evidence for a shared representation and allows to understand more in depth the translation process.

However, our system is a bit different, the decoders from the studied architecture are not the same. Then, we don't just visualise the output of the decoders, but the intermediate representations too and their corresponding words.

1. Overview of the app

The app contains 3 different visualization tools. These tools allow to obtain a better insight of the vectors processed in the Neural Machine Translation, especially in the intermediate part and in the decoder. I will introduce the third tool in the Discussion & Future Work part.

a. Technology Stack

Figure 13 - Technology Stack

The Technology Stack is the easiest to deal with data manipulation and data visualization. Python is very flexible and easy to use when it comes to data processing. Bokeh is a Python library of data visualization, it allows to create interactive dashboards easily. Flask is a Python web microframework, here it is used to embed the Bokeh dashboards on a web page.

b. Goals of the visualisation

During this project, we create 2 different viewers. Each of them has a precise goal and analyses a specific part of the architecture.

i. The Interlingua Viewer

The goal of the Interlingua Viewer is to, on the one hand, explore visually the intermediate representations of the sentences from the Machine Translation (MT) system and on the other hand, visualise the words vectors from these sentences. It is a very helpful visualisation tool since the aim of the MT system is to create an interlingua representation. With this tool, the researchers can identify the differences and evaluate the strength and the weakness of the model on the intermediate representation.

ii. The Decoder Viewer

The goal of this tool is to visualise and identify the most impactful layers during the decoding process and also to visualise the evolution of the differences between the different

language through the same decoder. With the Decoder Viewer, the researchers can have a better understanding of the decoding process since it is generally quite difficult to obtain feedbacks from a neural network.

c. Data Derivation

i. Interlingua data derivation

In order to visualise the sentence and the word embeddings, first a data preparation is needed. The goal is to obtain, at the end, a file with the link between each translation and the link between all the sentences with their words. The needed data were initially separated in 9 files which are the following :

- the sentences on a text file (*a file per language*)
- the embeddings of the sentence on a json file (*a file per language*)
- the words and their embeddings on a json file (*a file per language*)

To facilitate the visualization part, it's better to have a file with all the data and the links between each important information and since the data can grow, we want to pre-process as far as we can, so that in real-time the CPU is free for the visualization load.

The output is the following :

- a mapping from the sentences to the words and a link between the translations on a json file.

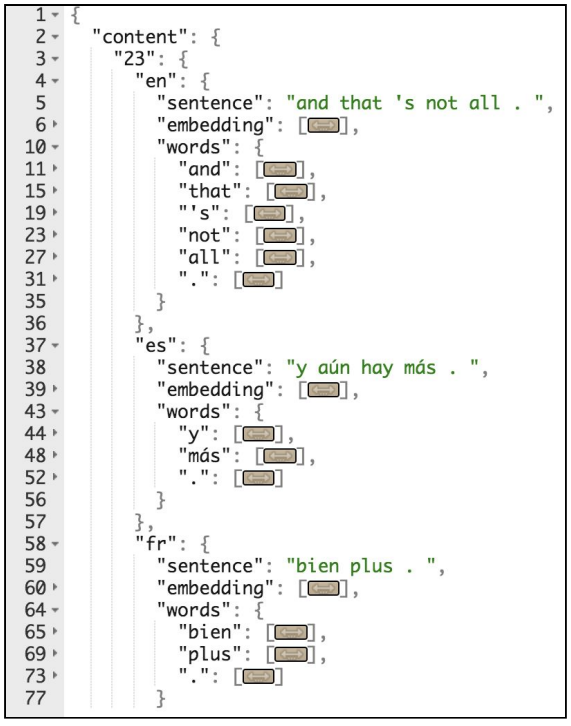


Figure 14 - Output file

Each translation is identified by an index (here, 23).
Then, the information is grouped by language :

- sentence : a string.
- embedding : a 2-dimension list.
- words : a dictionary
    - Each key is a word.
    - Each value is a 2-dim embedding.

The steps to obtain this output are the following :

1. Load the sentences and their embeddings for each language.
2. Link the sentences and the embeddings.
3. Load the list of words for each sentence by using SpaCy, a NLP library.
4. Apply UMAP on the embeddings.
   The dimension is reduced from 512 to 2.
5. Load the words and their embeddings for each language.
6. Apply UMAP on the embeddings (512 → 2).
7. Keep only the same sentences among all the languages.
8. Map the sentences to the words.

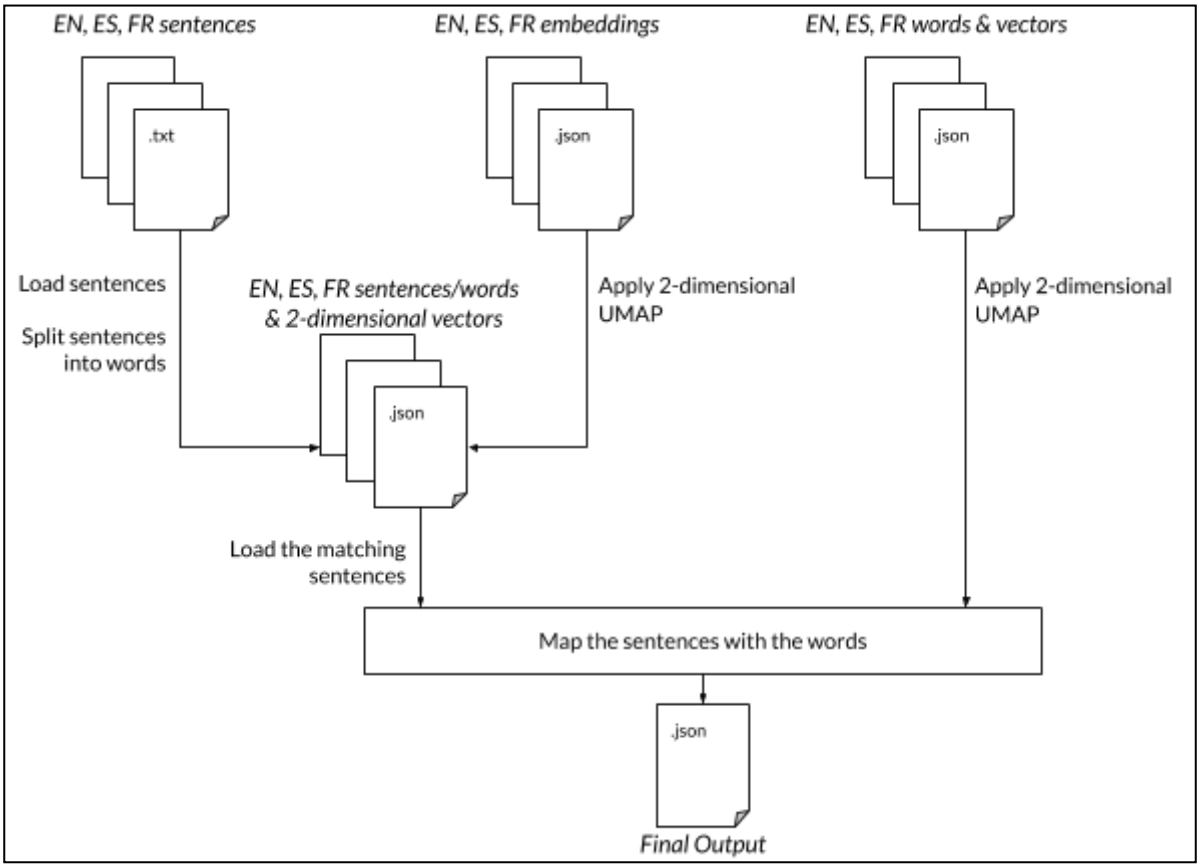


Figure 15 - Interlingua data derivation scheme

ii. Decoder data derivation

This data derivation is easier than the Interlingua one. The goal is to have the link for the 6 layers and each available translation between the sentences and the embeddings. It's very similar to the first steps from the previous paragraph. The inputs are the following :

- the sentences for each language (*same file as before*)
- the decodings for each layer on a json file (*same format as the embeddings above*) :
  In all, there are 30 decodings files :
  - the decodings to English for each layer : French to English, Spanish to English and English to English
  - the decodings to Spanish for each layer : English to Spanish and Spanish to Spanish. The current architecture isn't trained for French to Spanish.

It was not possible in this case to study the decodings to French, because this language was added after, as explained in the part 1.1.3 (Overview of the studied architecture).

There are 2 outputs, one for each decoder : the English decoder and the Spanish decoder.

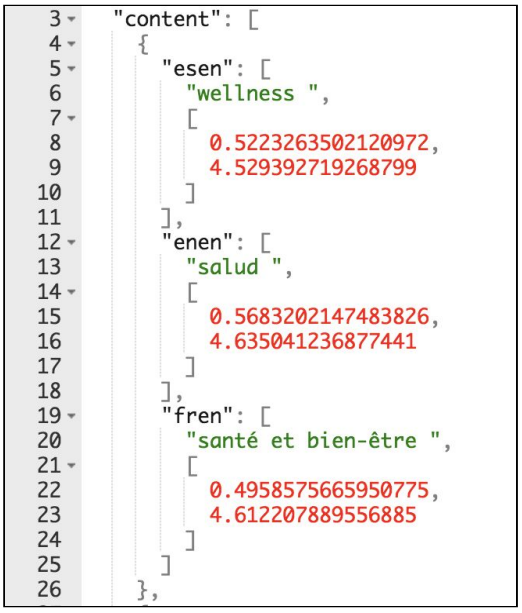


Figure 16 - English decoder file

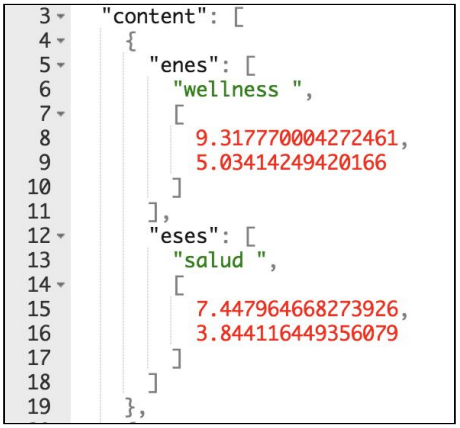


Figure 17 - Spanish decoder file

In "content", there is a list with each translation. Each element of the list stocks a dictionary with the 2 or 3 translations : Spanish to English (*esen*), English to English (*enen*), etc... The value of each key is a list with the sentence and the vector.

This data structure has been created before the Interlingua structure and is not as good as the previous one. However, since there are not too many sentences : 144 for the English decoder and 154 for the Spanish decoder, the processing time wasn't so bad and at this time, it was a satisfying structure.

In a nutshell, the steps to create one of these 2 files are the following :

1. Load the sentences for each language (3 files)
2. For each layer, load the decodings (3 or 2 files)
3. Keep only the common sentences
4. Then, apply UMAP on each vectors (512 → 2)
5. Link the sentences with the vectors and the translations

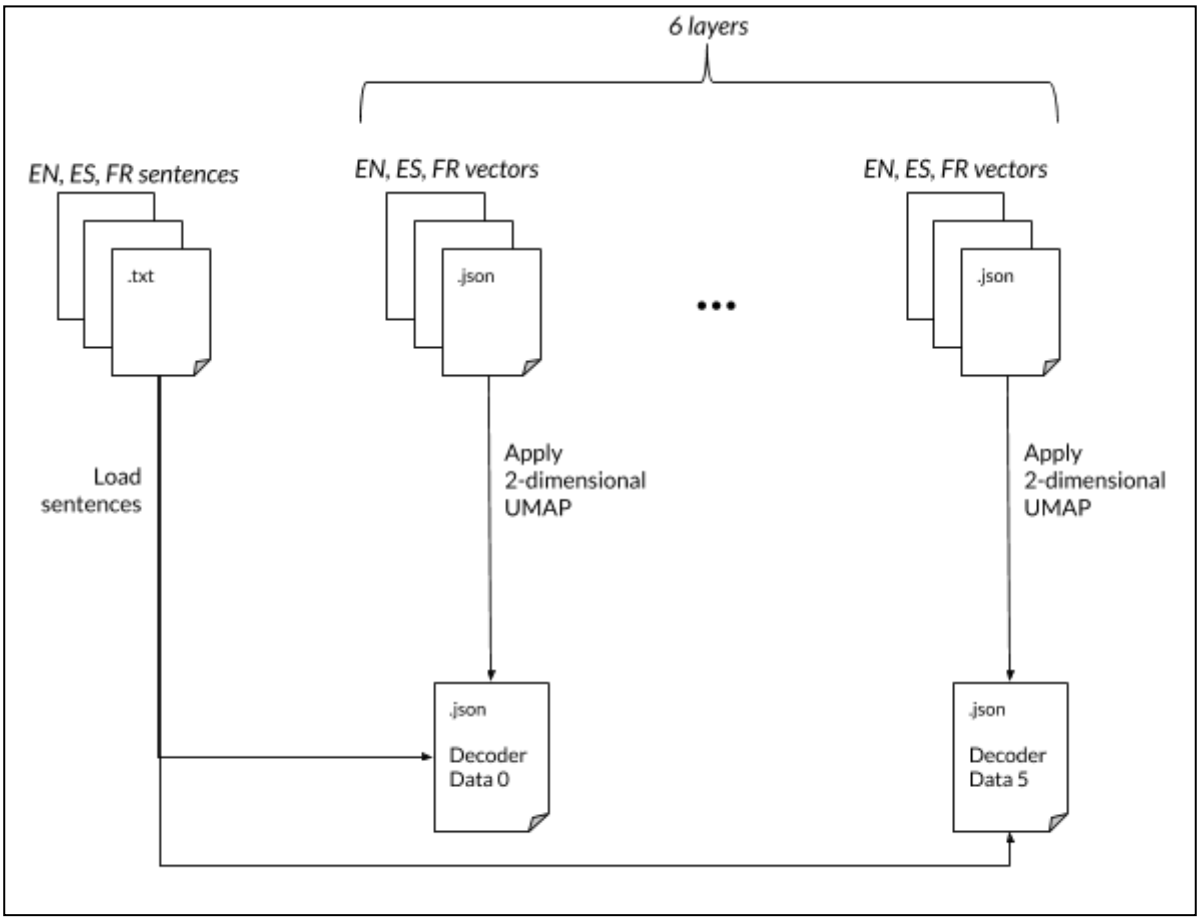


Figure 18 - Decoder data derivation scheme

d. Application overview

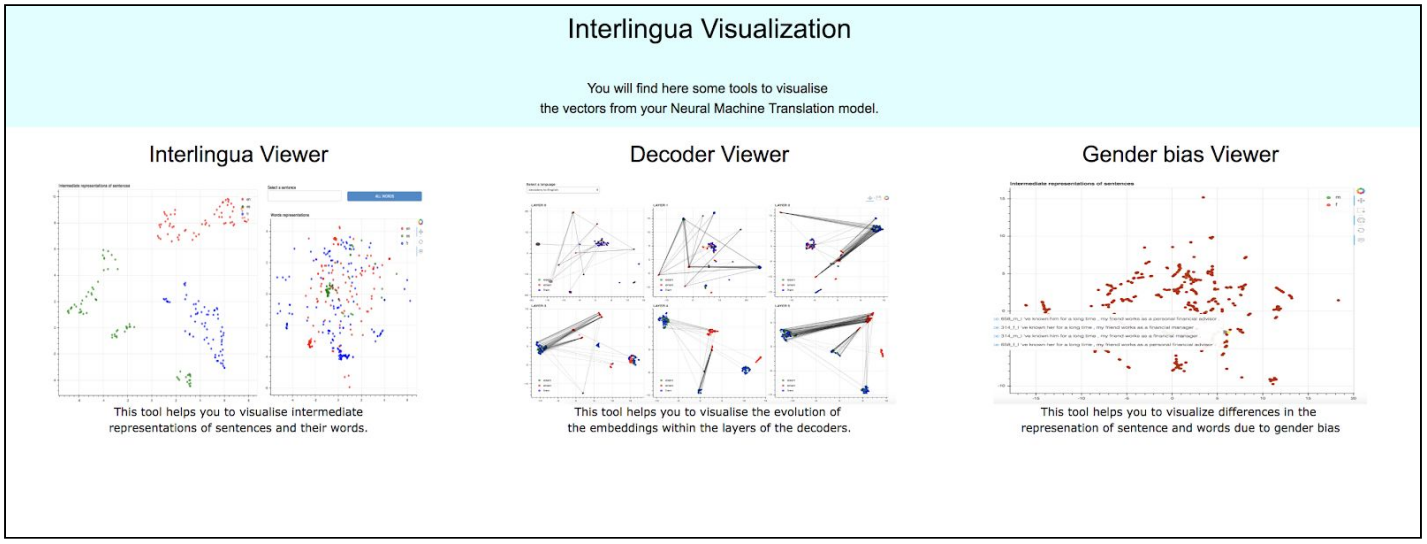


Figure 19 - Screenshot of the main page

All the visualization tools are displayed on the main page above. It is a basic web page which redirects to the Interlingua or the Decoder Viewer. Here, the user can have an overview on the different tools and a brief explanation on the use case. I will explain more in depth in the 2 following parts how to use these tools.

2. Interlingua viewer

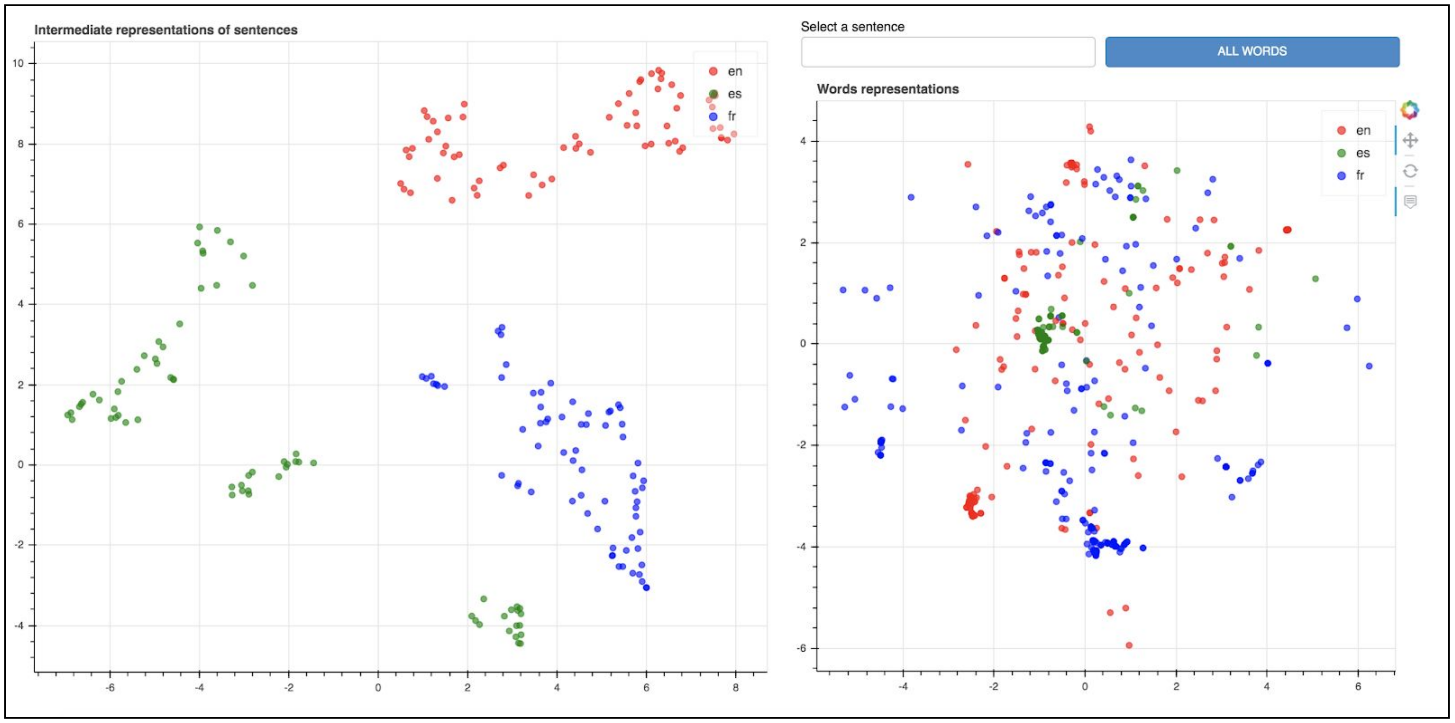


Figure 20 - Interlingua Viewer

The Interlingua Viewer is the first tool that we focused on during this project. Initially, we were interested in visualising the intermediate representations from the model in each language. The goal is here to determine how far these representations are and thus, obtain evidences on how the intermediate representations selected are close or not to a true interlingua.

Our second goal was to add to the viewer of the sentences a viewer of the words. This way, the user could obtain a more precise information about the difference between the translations of a sentence. With this fine grain detail, domain experts can gain understanding on where the differences come from.

a. Visual Design

Since we chose to apply a 2-dimensional UMAP on the data, the easiest and the clearest way to visualise the vectors was to create 2 scatter plots. This app is composed of 2 coordinated views.

The embeddings are categorized on the plots according to the language. Each color represents a different language. Of course, the colour coding is the same on the sentence plot and on the word plot, in this way, the user can understand easily the link between the 2 plots.

In order to clearly identify the sentence the user is focusing on, there are some specifications on the selection. By selecting a point on the sentence plot, this selected sentence will keep the same color while the others points will become semi-transparent.

b. Interaction Design

In this part, I will explain more in depth how the interactions behave.

i. Intermediate representations of sentences

The left plot displays the intermediate representations of the sentences. Here, the model used takes into account 3 languages, but it could be more.

As it can be seen on the screenshot below, the points are separated by language :

- en for English
- es for Spanish
- fr for French

By hovering over a point, the user can see the text of the sentence and the links with the other languages : the 2 lines point on the corresponding translations.

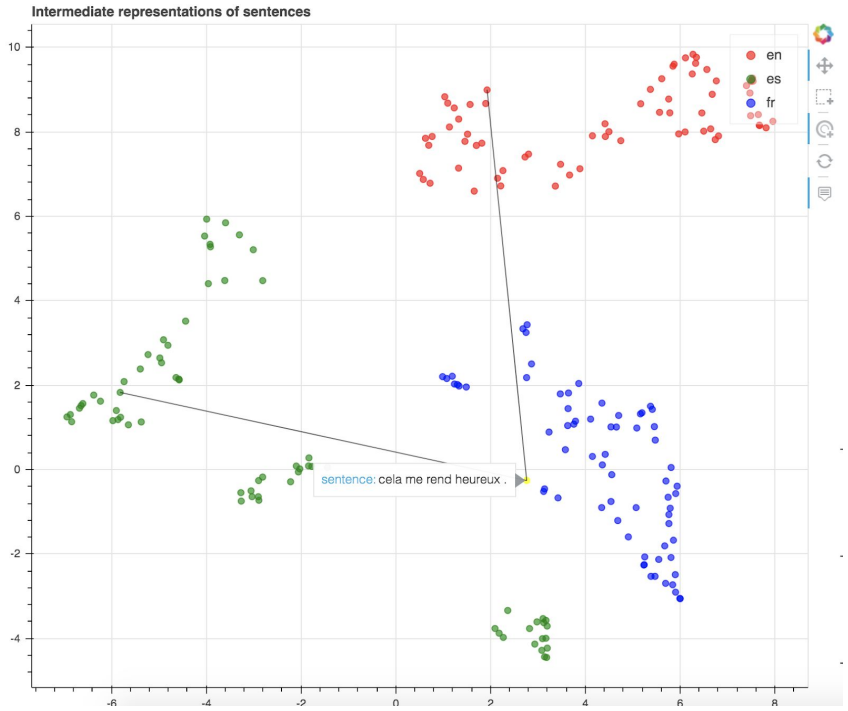


Figure 21 - Hovering over a sentence

The user can also click on a sentence to display further information : the words of this sentence and the same words in the other languages. There is 2 ways to zoom on a sentence :

- by clicking on a point

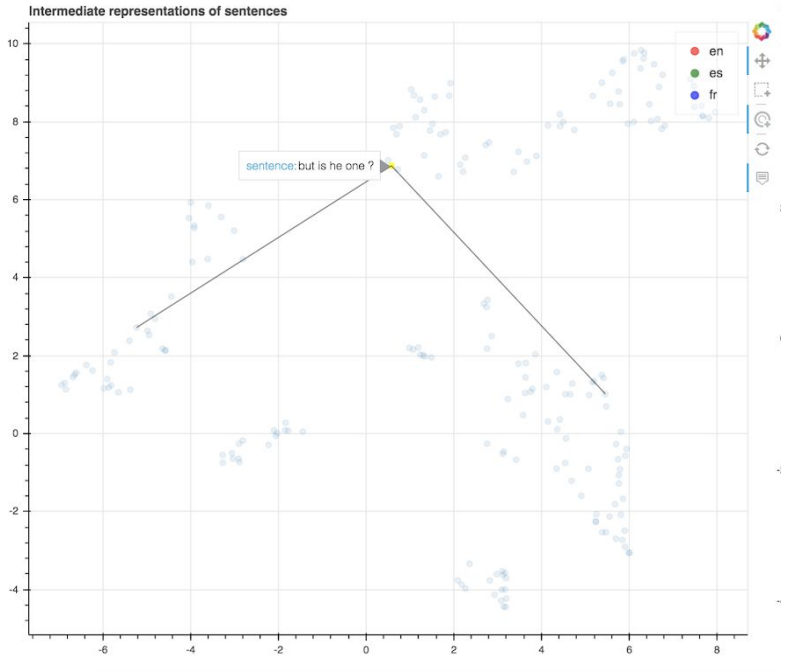
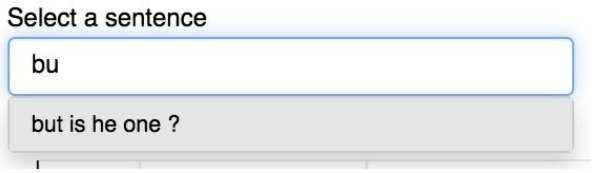


Figure 22 - Clicking on a sentence

- by writing the sentence in the search bar just above the words representation on the board :



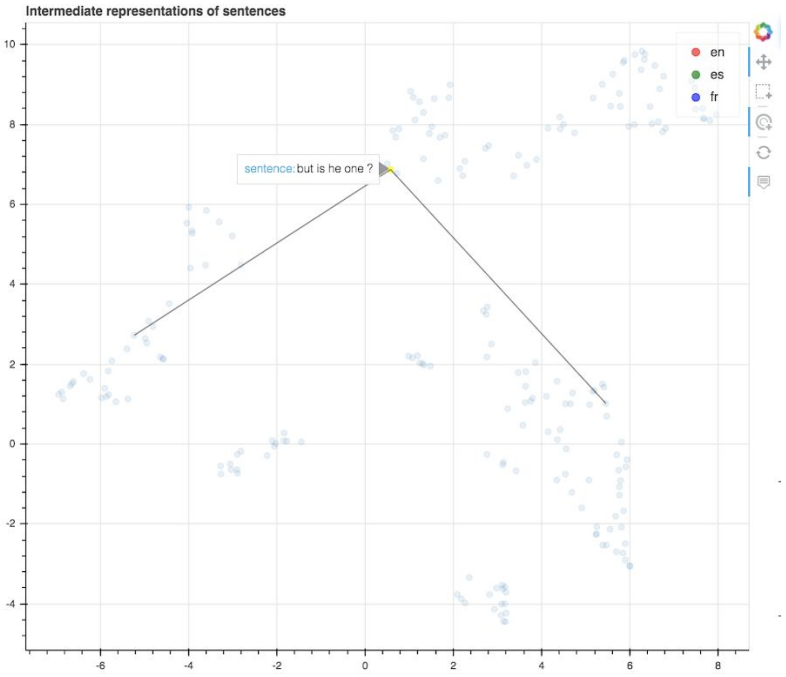


Figure 23 - Searching a sentence

The search bar has an autocomplete feature. This way, once the user has written at least 2 characters, he will be able to see all the sentences, no matter which language, with this sequence of letters (or numbers). To highlight the target sentence, the user has to click on the right suggestion, he can't use the arrow keys (Bokeh doesn't support the key events).

ii. Representations of words

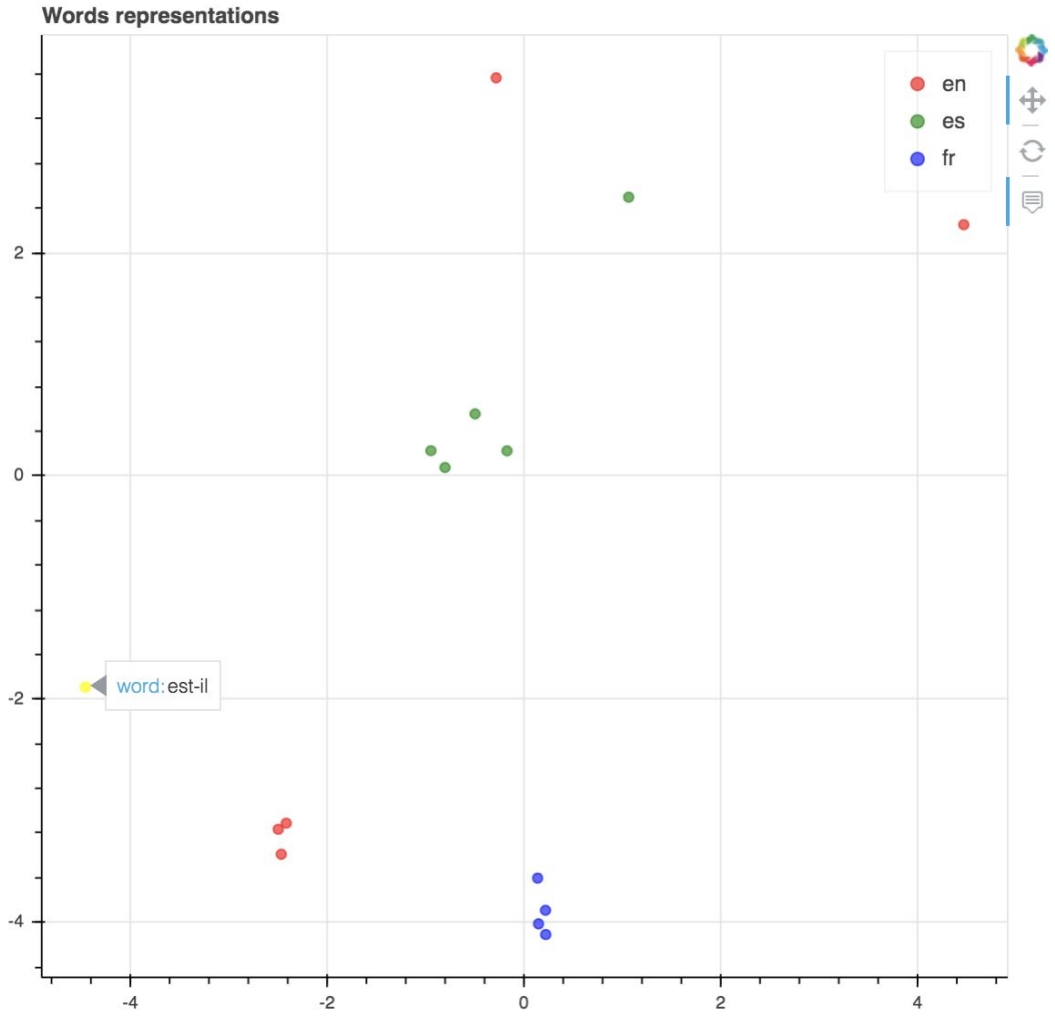


Figure 24 - Words representations

The plot from Figure 24 represents all the words initially, and once the user selects a sentence by clicking or searching, only the words from this sentence and its translations remain on the plot.

By hovering over a point, the user can see the text of the word, it's the same than with the sentences.

iii. Link between sentences and words

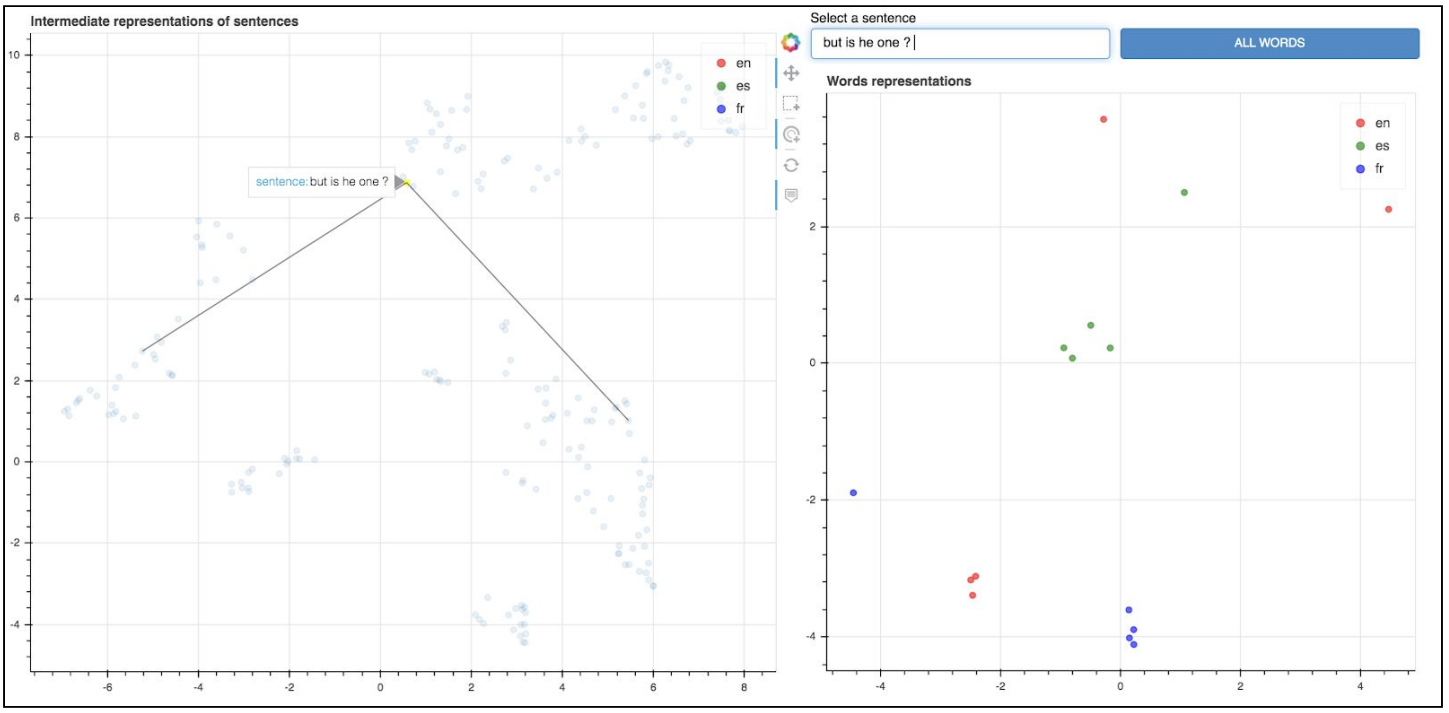


Figure 25 - Updating words representations

Once the user selects a sentence, he can see directly on the dashboard the corresponding words.

This plot has 3 possible states :

- Initially, the plot contains all the words.
- If the user clicks on a point from the sentence plot, the word plot will instantly update and display the words from the selected sentence and its translation.
- If the user doesn't select any sentence, the right plot will become a blank plot.

If the user wants to display the initial state, all the words, he has to click on the blue button "ALL WORDS" and the plot will be the same as at the beginning.

3. Decoder layers viewer

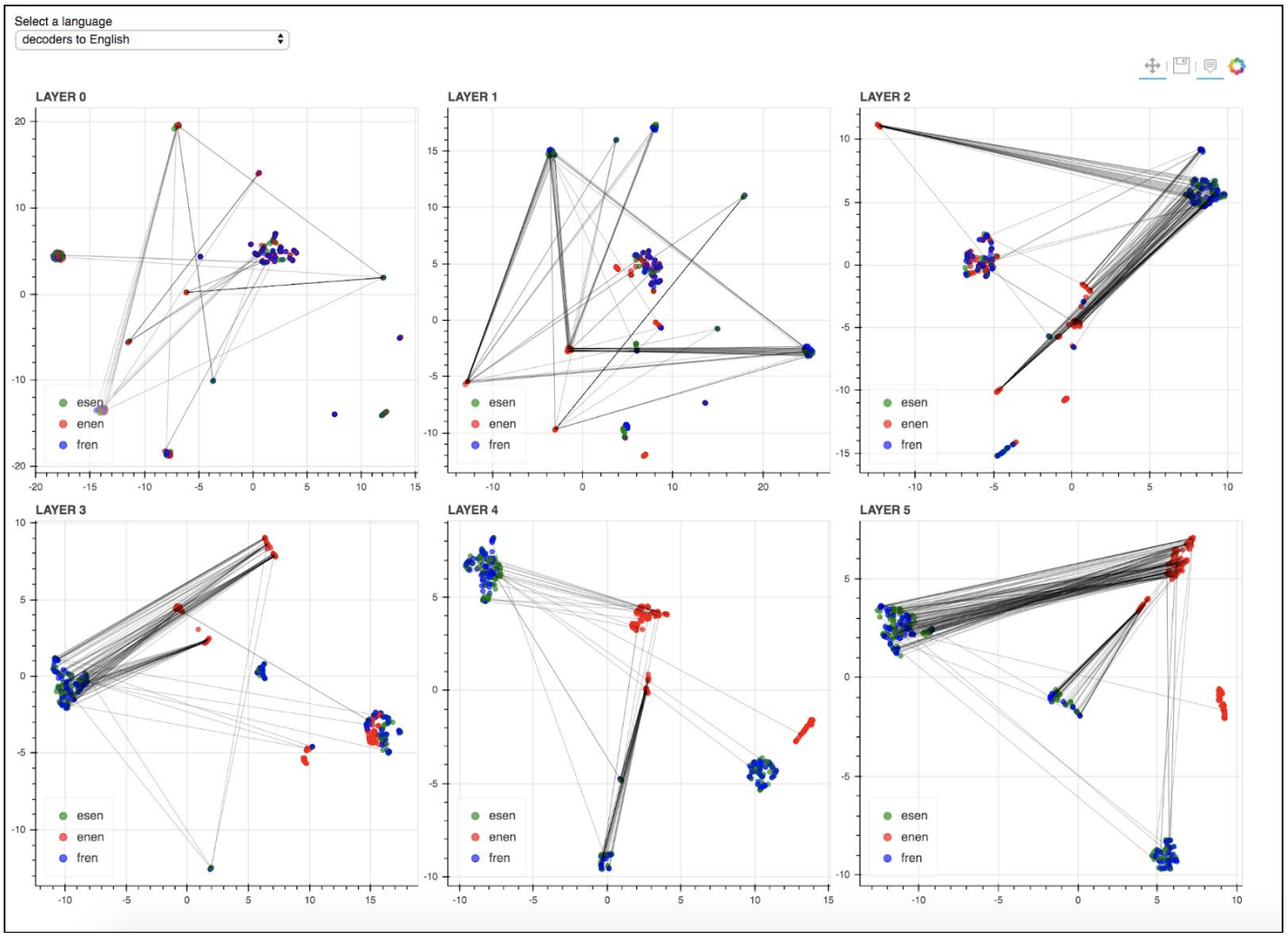


Figure 26 - Decoder Viewer

The idea of the Decoder Viewer came later. By analyzing the intermediate representations, we agreed that it would be interesting to visualise the evolution of these vectors in the decoder. Actually, if we generate all the data, it would be interesting to visualise too the encoder layers. This approach may provide us an idea on which parts of the decoder are doing the main translation work and on which parts seem to bring the representations of the different languages together.

a. Visual Design

This app is composed of 6 synchronized views organized in a small-multiples design. Indeed, we made the choice to have a different plot for each layer. The layer 0 plot represents the output vectors from the first layer of the decoder and then the layer 5 represents the vector of the translated sentence, since the decoder contains 6 layers.

As with the Interlingua Viewer, the data are 2-dimensional and scatter plots seem to be the best choice.

Here, we chose to compare on both side the link between the different languages and the evolution of this set through the layers. The vectors are categorized on a single plot by its initial language. Basically, the user can visualise on the one hand the evolution of the 3 possible languages to English, and on the other hand, the evolution of the 2 possible languages English and Spanish to Spanish for the Spanish Decoder.

b. Interaction Design

i. Layer representation

On each plot, we can display the sentence by hovering. In order to emphasize the distances between the translations and to have a better insight of the evolution, the link between the most dissimilar are displayed on the plots. By hovering over the lines, the user can obtain the cosine distance value computed by SciPy.

The measure given by SciPy is computed in this way :

$$1 - \frac{u \cdot v}{||u||_2 ||v||_2}$$

On the plots, only the distances superior to 1 are displayed, i.e. all the negative cosine values.

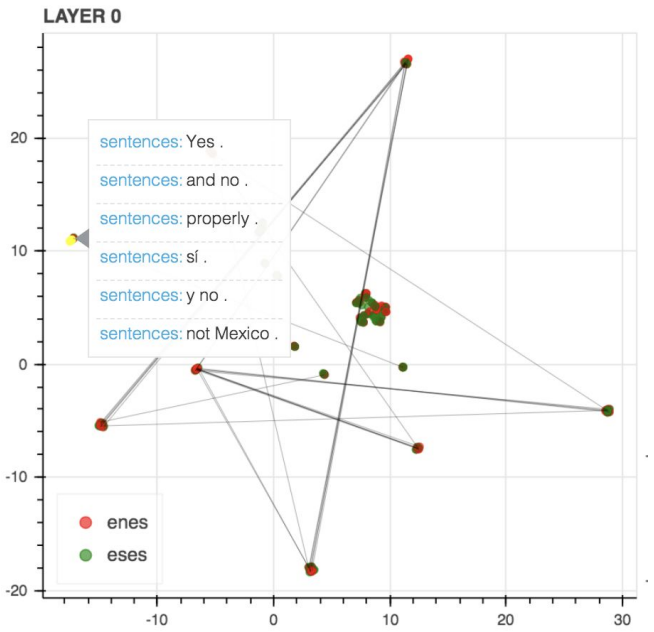


Figure 27 - Hovering over a sentence

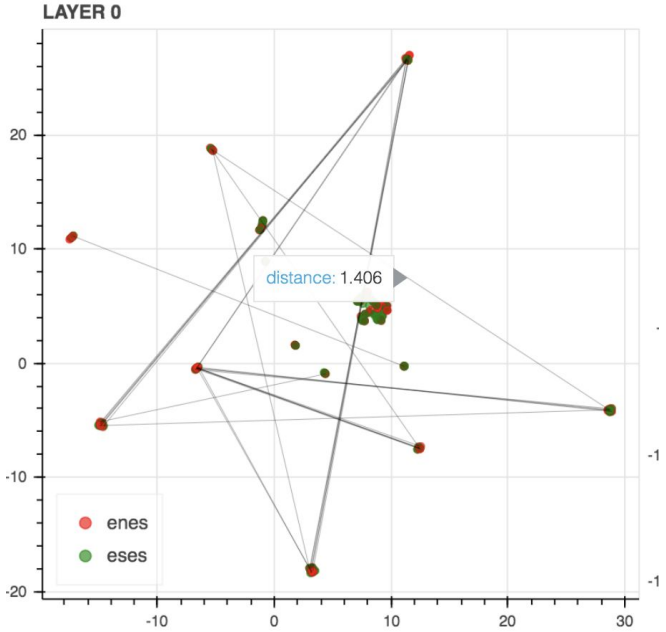
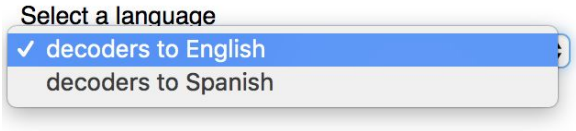


Figure 28 - Hovering over a line

ii. Switching to another decoder

Initially, the decoder represented on the dashboard is the English decoder. However, the user can switch to another decoder by using the selection tool at the top of the page. Of course, if the user has the needed data for other decoders, they could be add to the selection and then, display on the dashboard.

To change the data, the user needs to use the selection tool. Here, he can switch to the Spanish decoder.



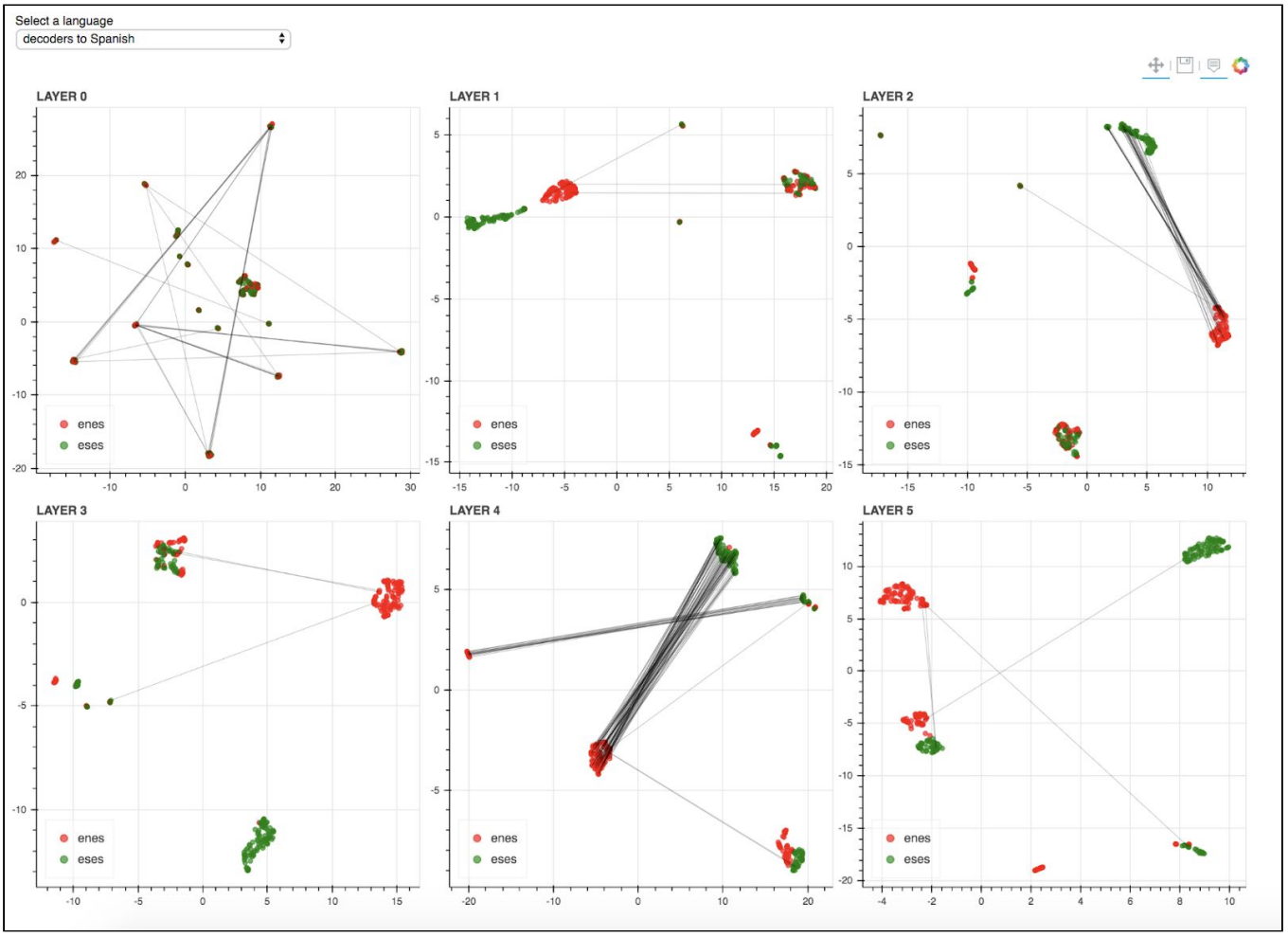


Figure 29 - Decoder Viewer with Spanish data

Once the user changes the data, the plots are updated and he can highlight the sentences or the distances.

The diskette icon above the plots on the right allows to save these plots as png files.

In this part, I will introduce 2 different use cases for the visualisation tools and how researchers could use these tools for their work.

1. Multilingual common representation in translation

There are mainly 2 different types of Machine Translation architecture :

- A unique pair of encoder-decoder for all the languages
- Multiple pairs of encoder-decoder for each language

In this 2 cases, there is an intermediate representation for each language and these representations should be close since they share the same meaning.

By using the Interlingua Viewer, researchers can display these representations and compare them by language.

Here, the intermediate representations from the Autoencoder-based Machine Translation system, introduced in the Background part, are used. The encoders-decoders are independent but the interlingua space is forced. The model is trained with 15 million translations in English, Spanish and French. For the visualisation, only the 130 sentences from the validation set are displayed. These 130 sentences are translated in the 3 languages.

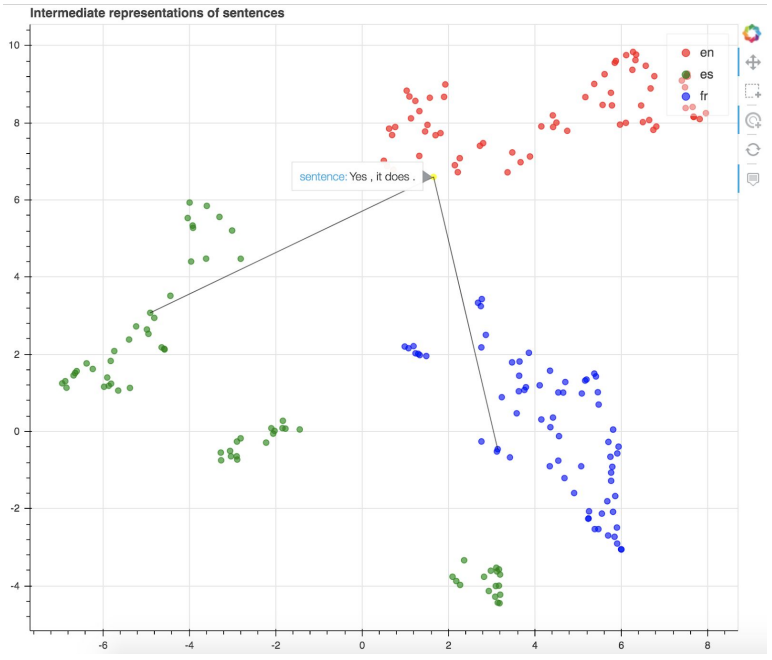


Figure 30 - Intermediate representations of sentences

In Figure 30, when the user explores a sentence, he can obtain the link with the translations. Thus, he can get easily visual information from the model.

2. Multilingual layer interpretation in translation decoding

The encoders and decoders used in Machine Translation are generally composed of multiple layers. The role of each layer is difficult to identify. In order to have a better understanding of the system, one of the best way is to visualise it.

With the Decoder Viewer, researchers can study each step of the decoding process.

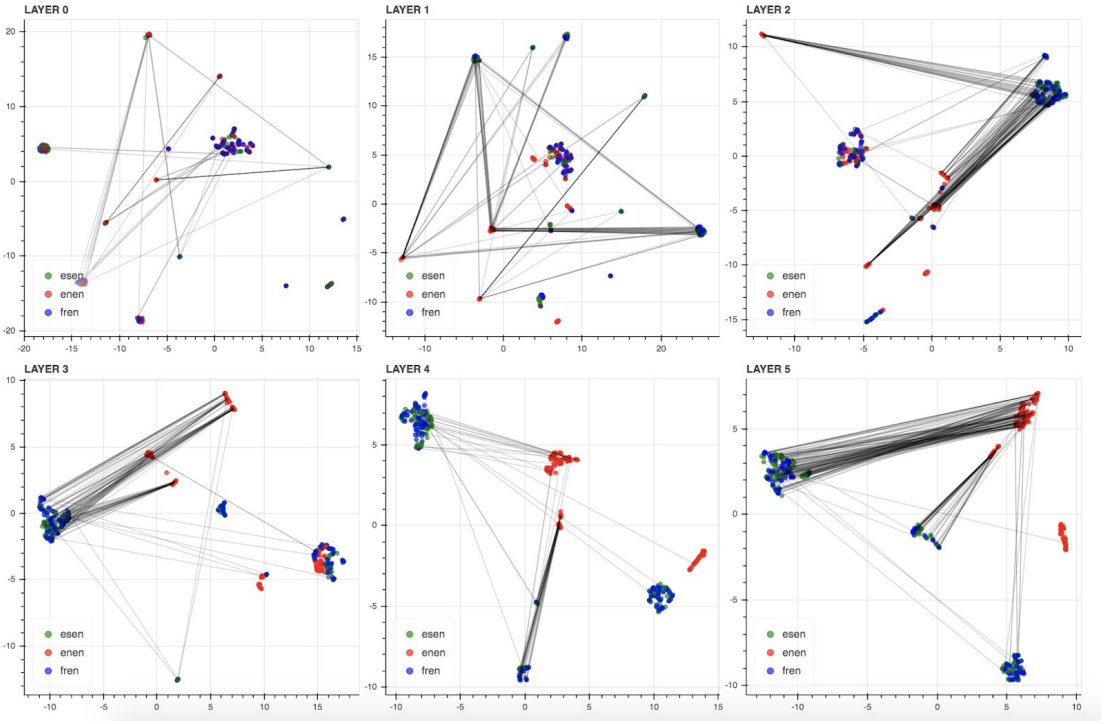


Figure 31 - Layers representations

Studying the evolution of the vectors, it is possible to identify where is the main translation work, which layer has the highest semantic implications, or which parts bring the more of parallel sentences together. In Figure 31, sentences become more grouped in the last layers.

On each view, the user can also study the links between the different languages.

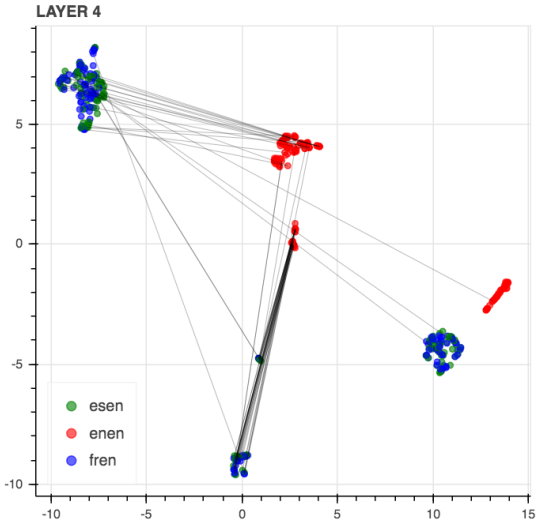


Figure 32 - Single layer representation

1. Visualise and compare other models

During this project, we focused essentially on a specific Machine Translation architecture. However, the visualisation tools are enough generic for being used in many different use cases, and not only for Machine Translation. Actually, this tool could be used for any Machine Learning system and then provide an insight of the model. Visualisation tools are one of the best manners for experts to understand complex systems.

The visualisation tools could be applied not only for multilingual representations. Since the colored categories depend on the input data, the vectors could be compared on other criteria than languages. By changing the input data, researchers identify another interesting use case for the Interlingua Viewer.

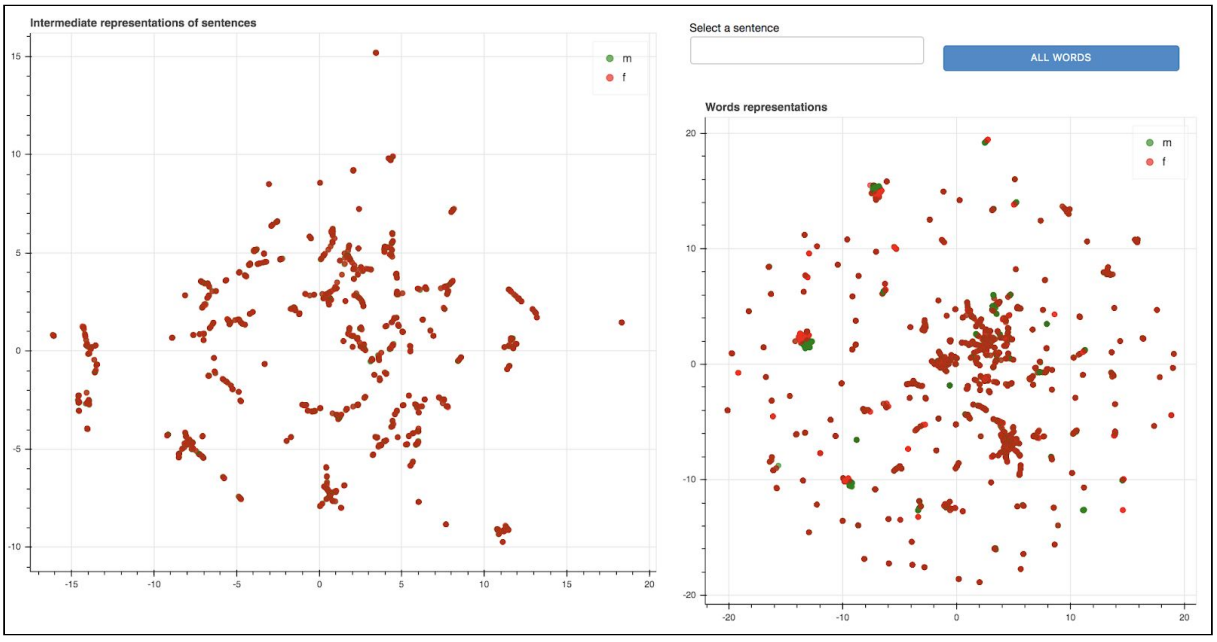


Figure 33 - Gender bias Viewer

In Figure 33, the displayed data are categorized by gender. It includes 1019 parallel sentences in the feminine and the masculine forms. These embeddings are generated by using a Contextual Word Embeddings system, ELMo [12]. The interest is that same words can be compared according to the context of a sentence.

All the sentences follow the same format :

I've known *her/him* for a long time, my friend works as a *neutral profession*.
(e.g. firefighter)

Then, by selecting a sentence, the words representations are updated and the mentioned profession is highlighted in the 2 genders. In Figure 34, we can notice a small different between the 2 words.

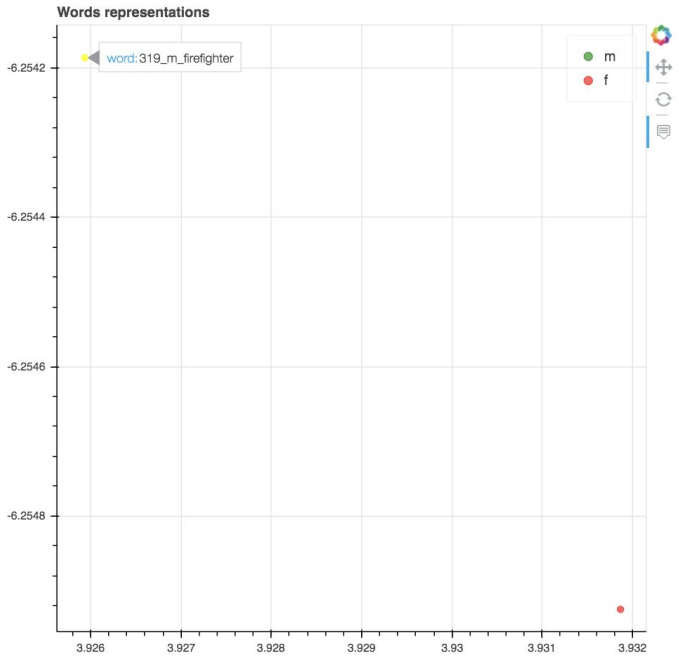


Figure 34 - Word representation for firefighter

Thus, this visualisation tool can be easily used for identifying gender bias in Contextual Word Embeddings.

We could also for instance categorized the sentences by Machine Translation models on the plot and then get a great comparison of the different systems.

A tool such as the Decoder Viewer could have also more than one use case. First, it could work as well with Encoder outputs. Furthermore, since a lot of architectures, not only for Machine Translation, used an Encoder-Decoder system, we could easily imagine that experts could use the viewer for comparing the layers and understanding the role of each of them.

The Contextual Word Embeddings model BERT [13] is based on an architecture with several encoder layers, essentially 12 or 24. Basically, Words Embeddings can be retrieved from each layer or by summing different layer outputs. Visualising the outputs of each layer could be an interesting approach.

2. New interesting features

The tool is open-source and is available at https://upc-nmt-vis.herokuapp.com/ or on Github at https://github.com/elorala/interlingua-visualization.

In the future, we can imagine a lot of new features to improve the user experience.

a. Upload new data

One the most interesting feature would be to provide a way to upload its own data standardized according to the appropriate format. Since the app is deployed on a server, it would be easier than changing directly the files from the source code and running the app on a localhost. It's the only way from now to use different data.

b. Gather more information

Another feature would be to give access to more information such as attention value from the model or distances value between translations. Of course, it would add more data derivation steps.

It would be also interesting to allow the researchers to have access to different dimensionality reduction at the same time, t-SNE and UMAP for instance.

CONCLUSION

This thesis has introduced visualisation tools for multilingual intermediate representations at the level of sentences and words and for multilingual sentences in decoder layers. We have highlighted that this tool is extremely adaptable to different data and to different use cases. Indeed, the Gender bias Viewer has been created directly from the Interlingua Viewer only by changing the input data. Thus, in the future, the application could be used for several different use cases. With these tools, we allow researchers and experts to visualise and to get an insight of their Machine Learning systems. The application already helps a research team in the understanding of their model.

REFERENCES

[1] Escolano, C., Costa-jussà, M. R., & Fonollosa, J. A. R. (2018). *(Self-Attentive) Autoencoder-based Universal Language Representation for Machine Translation*. Retrieved from http://arxiv.org/abs/1810.06351

[2] Introduction to Autoencoders
https://www.jeremyjordan.me/autoencoders/

[3] Dorr, B., & Hovy, E. (n.d.). Encyclopedia of Language and Linguistics, (ELL2). Machine Translation: Interlingual Methods. *Lamp.Cfar.Umd.Edu*, 1–20. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Natural+Language+Processing+and+Machine+Translation+Encyclopedia+of+Language+and+Linguistics+,+2nd+ed+.+(+ELL2+).+Machine+Translation+:+Interlingual+Methods#0%5Cn http://scholar.google.com/schol

[4] McInnes, L., Healy, J., & Melville, J. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. Retrieved from http://arxiv.org/abs/1802.03426

[5] Performance Comparison of Dimension Reduction Implementations
https://umap-learn.readthedocs.io/en/latest/benchmarking.html

[6] 1st Workshop on Visualization for AI Explainability,
co-located with IEEE Visualization,
Berlin, October 2018 (https://visxai.io/).

[7] The Beginner's Guide to Dimensionality Reduction

https://idyll.pub/post/visxai-dimensionality-reduction-1dbad0a67a092b007c526a45/

[8] TensorFlow Playground
https://playground.tensorflow.org/

[9] Smilkov, D., Thorat, N., Nicholson, C., Reif, E., Viégas, F. B., & Wattenberg, M. (2016). *Embedding Projector: Interactive Visualization and Interpretation of Embeddings*. (Nips). Retrieved from http://arxiv.org/abs/1611.05469

[10] Embedding Projector
https://projector.tensorflow.org/

[11] Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., … Dean, J. (2016). *Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation*. 1–16. Retrieved from http://arxiv.org/abs/1611.04558

[12] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations*. Retrieved from http://arxiv.org/abs/1802.05365

[13] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. (Mlm). Retrieved from http://arxiv.org/abs/1810.04805

LIST OF FIGURES