Deconvolution : a deep tool box for linguistic analysis

Anonymous ACL submission

Abstract

This document contains the instructions for preparing a camera-ready manuscript for the proceedings of ACL 2018. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used for both papers submitted for review and for final versions of accepted papers. Authors are asked to conform to all the directions reported in this document.

1 Introduction

As in many other fields of data analysis, Natural Language Processing (NLP) has been strongly impacted by the recent advances in Machine Learning, more particularly with the emergence of Deep Learning techniques. These techniques outperform all other state-of-the-art approaches on a wide range of NLP tasks and so they have been quickly and intensively used in industrial systems. Such systems rely on end-to-end training on large amounts of data, making no prior assumptions about linguistic structure and focusing on stastically frequent patterns. Thus, they somehow step away from computational linguistics as they learn implicit linguistic information automatically without aiming at explaining or even exhibiting classic linguistic structures underlying the decision.

This is the question we raise in this article and that we intend to address by exhibiting classic linguistic patterns which are indeed exploited implictly in deep architectures to lead to higher performances. Do neural networks make use of co-occurrences and other standard features, considered in traditional Textual Data Analysis (TDA)? Do they also rely on complementary linguistic structure which is invisible to traditional techniques? If so, projecting neural networks features back onto the input space would highlight new linguistic structures and would lead to improving the analysis of a corpus and a better understanding on where the power of the Deep Learning techniques comes from.

Our hypothesis is that Deep Learning is sensitive to the linguistic units on which the computation of the key statistical sentences is based as well as to phenomena other than frequency and complex linguistic observables. The TDA has more difficulty taking such elements into account – such as linguistic linguistic patterns (Mellet et Longrée, 2009). Our contribution confronts Textual Data Analysis and Convolutional Neural Networks for text analysis. We take advantage of deconvolution networks for image analysis in order to present a new perspective on text analysis to the linguistic community that we call deconvolution saliency. Our deconvolution saliency corresponds to the sum over the word embedding of the deconvolution projection of a given feature map. Such score provides a heat-map of words in a sentence that highlights the pattern relevant for the classification decision. We examine z-scoring and deconvolution saliency on three languages: English, French and Latin. For all our datasets, deconvolution saliency highlights new linguistic observables, invisible with z-scoring alone.

2 Related work

Convolutional Neural Networks (CNNs) are widely used in the computer vision community for a wide panel of tasks: ranging from image classification, object detection to semantic segmentation. It is a bottom-up approach where we applied an input image, stacked layers of convolutions, non-linearities and sub-sampling. Encouraged by the success for vision tasks, researchers applied CNNs

to text-related problems. The use of CNNs for sentence modeling traces back to (Collobert and Weston, 2008). Collobert adapted CNNs for various NLP problems including Part-of-Speech tagging, chunking, Named Entity Recognition and semantic labeling (cite). CNNs for NLP work as an analogy between an image and a text representation. Indeed each word is embedded in a vector representation, then several words build a matrix (concatenation of the vectors).

We first discuss our choice of architectures. If Recurrent Neural Networks (*mostly GRU and LSTM*) are known to perform well on a broad range of tasks for text, recent comparisons have confirmed the advantage of CNNs over RNNs when the task at hand is essentially a keyphrase recognition task [1].

In Textual Mining, we aim at highlighting linguistics patterns in order to analyze their constrast: specificities and similarities in a corpus (Feldman, R., and J. Sanger, 2007; L. Lebart, A. Salem and L. Berry, 1998). It mostly relies on frequential based methods such as z-scoring. However, such existing methods have so far encountered difficulties in underlining more challenging linguistic knowledge, which has yet been empirically observed as for instance syntactical motifs (Mellet and Longrée, 2009).

In that context, supervised classification, especially CNNs, may be exploited for corpus analysis. Indeed, CNN learns automatically parameters to cluster similar instances and drive away instances from different categories. Eventually, their prediction relies on features which inferred specificities and similarities in a corpus. Projecting such features in the word embedding will reveal relevant spots and may automatize the discovery of new linguistic structure as the previously cited, syntactical motifs. Moreover, CNNs hold other advantages for linguistic analysis. They are static architectures that, according to specific settings are more robust to vanishing gradient, and thus can also model long-term dependency in a sentence (Dauphin et al., Wen et al. 2016, and Adel and Schutze). Such a property may help to detect structures relying on different parts of a sentence.

All previous works converged to a shared assessment: both CNNs and RNNs provide relevant, but different kinds of information for text classification. However, though several works have studied linguistic structures inherent in RNNs, to our knowledge, none of them have focused on CNNs. A first line of research has extensively studied the interpretability of word embeddings and their semantic representations (cite). When it comes to deep architectures, Krizhevsky et al. (cite) used LSTMs on character level language as a testbed. They demonstrate the existence of long-range dependencies on real word data. Their analysis is based on gate activation statistics and is thus global. On another side, Li et al. (cite) provided new visualization tools for recurrent models. They use decoders, t-SNE and first derivative saliency, in order to shed a light on how neural models work. Our perspective differs from their line of research, as we do not intend to explain how CNNs work on textual data, but rather use their features to provide complementary information for linguistic analysis.

Although the usage of RNNs is more common, there are various visualization tools for CNNs analysis, inspired by the computer vision field. Such works may help us to highlight the linguistic features learned by a CNN. Consequently, our method takes inspiration from those works. Visualization models in computer vision mainly consist in inverting latent representations in order to spot active regions or features that are relevant to the classification decision. One can either train a decoder network or use backpropagation on the input instance to highlight its most relevant features. While those methods may hold accurate information in their input recovery, they have two main drawbacks: i) they are computationally expensive: the first method requires training a model for each latent representation, and the second relies on backpropagation for each submitted sentence. ii) they are highly hyperparameter dependent and may require some finetuning depending on the task at hand. On the other hand, Deconvolution Networks, proposed by Zeiler et al (cite) ?, provide an off-the-shelf method to project a feature map in the input space. It consists in inverting each convolutional layer iteratively, back to the input space. The inverse of a discrete convolution is computationally challenging. In response, a coarse approximation may be employed which consists of inverting channels and filter weights in a convolutional layer and then transposing their kernel matrix. More details of the deconvolution heuristic are provided in section **??**. Deconvo-

lution holds several advantages. First, it induces minimal computational requirements compared to previous visualization methods. Also, it has been used with success for semantic segmentation on images: in ?; Noh et al (cite) demonstrate the efficiency of deconvolution networks to predict segmentation masks to identify pixel-wise class labels. Thus deconvolution is able to localize meaningful structure in the input space.

3 Model

3.1 CNN for Text Classification

We propose a deep neural model to capture linguistics patterns in text. This model is based on Convolutional Neural Networks with an embedding layer for word representations, one convolutional with pooling layer and non-linearities. Finally we have two fully-connected layers. The final output size corresponds to the number of classes we attempt to train. The model is trained by cross-entropy with an Adam optimizer. Figure 2 shows the global structure of our architecture. The input is a sequence of words $w_1, w_2 ... w_n$ and the output contains class probabilities (for text classification). The embedding is built on top of a Word2Vec architecture trained on a Skip-gram model. Our text tokenizer keeps all the words to make sure all linguistic material is detected at the end by the model. This embedding is also finetuned by the model to attain optimal text-classification accuracy.

The Convolutional layer is based on a two-dimensional convolution, the same as used for image convolution, but with a fixed width equal to the embedding size. Indeed, word embeddings are not intended to hold spatial information. With this setting, our usage of the two-dimensional convolution is in reality the same as a one-dimensional convolution (the default settings for text). The only parameter we adjust here is the height of the filter corresponding to the number of words we want to put in the filter.

they can be easily parallelized and may also be be easily used by the CPU, which is a practical solution for avoiding the use of GPUs at test time and thus providing wider access to our tools.

3.2 Deconvolution

Since we use same architecture as image detection, making a deconvolutional layer is really straightforward. There are several methods to visualize the deep internal mecanisms of a neural network. One is known as convolutional transposed. Our deconvolutional network use the same embedding and convolution layer as we use for the classification but we replace the finale dense layer by a transposed convolution layer. After we trained the model we setup the weight of each neuron of the deconvolutional network with the learned weights of the classification network. The result is a new network that takes as input a sequence of words and gives as output all the trained filters of the text classification applied on the given sequence. Then the activation score of each word is calculated as shown in Equation 1 with x is the size of the embedding, and y the number of applied filters :

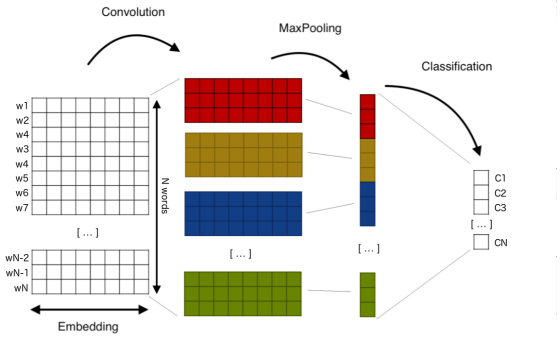

Figure 1: CNN model

$$\sum_{i=1}^{x}\sum_{j=1}^{y} a_{ij} = s_n \quad (1)$$

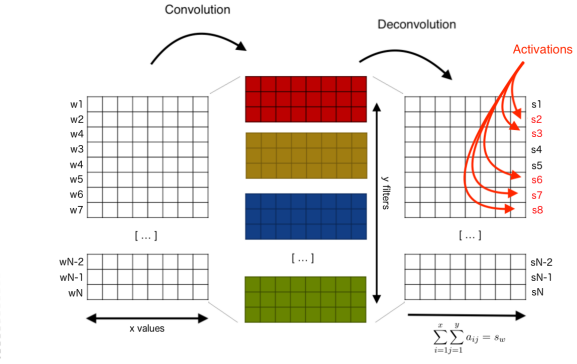

Figure 2: Deconvolution model

With this method we are able to show a sort of topology of a sequence of words. All words have an unique activation score related to the others. We will see now that this output of the de-

	z-score	Deep Learning
Latin	84%	93%
French	89%	91%
English	90%	97%

Figure 3: Prediction task acuray with z-score and Deep Learning

convolution gives us much information on how the network makes its final descision (prediction). There are well known linguistic marks encoded inside the networm, as well as more complex patterns based on co-occurrences and possibly also on grammatical and syntaxic analysis.

4 Experiments

4.1 Z-score Versus Activation-score

Z-score is one of the most used methods in linguistic statistics. It compares the observed frequency of a word with the frequency expected in the case of a "normal" distribution. This calculation readily gives, for example, the most specific vocabulary of a given author in a contrastive corpus. The highest z-scores are the most specific words in this case. This is a simple but strong method for analyzing features of text. It can also be used to classify word sequences according to the global z-score (sum of the score) in the sequence. The mean accuracy of this method on our data set is around 87%, which confirms z-score is in fact meaningful on contrastive data. On the other hand, most of the time deep learning attains greater than 90% accuracy in text classification (As shown on Figure 3 – a benchmark of our three datasets detailed in the next subsections 4.2,4.3,4.4).

This means that the training methods can learn also on their own some of the linguistic specificities useful in distinguishing between classes of text or authors. We've seen in work on images that thi is the role of convolution. It learns an abstraction of the data to make classification easier. The question is: what is the nature of this abstraction on text? We will see now that deep learning detects automatically words with hight z-score but apparently this is not the only linguistic structure detected.

To make the two values comparable, we normalize them with a maximum score of around $+38$ and a minimum of -38. This interval gives two thresholds for the z-score: over 2 a word is considered as specific and over 5 it's strongly specific (and the oposite with negative values). For the activation score it's just a matter of activation strength.

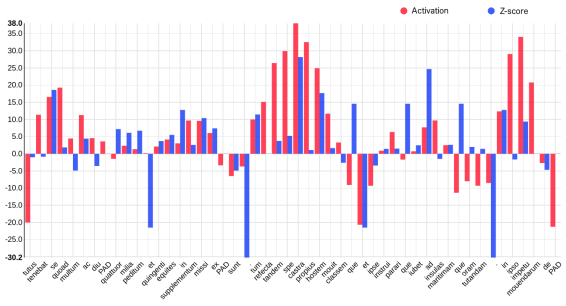

Figure 4: Z-score versus Activation-score

The Figure 4 shows us a comparison between z-score and activation-score on a sequence extract form our Latin corpora (Livy Book XXIII Chap. 26). Here it's an example of specific word use by Livy[1]. As we can see, when the z-score is the highest there is a sort of activation spike (for example around the word *castra*). However, this is not always the case: for example small words as *que*, *ad* and *et* are also high in z-score but they do not activate the network at the same level. We saw in (reference ****) that deeplearning is more sensitive to long words, but we can see also on Figure 4 that words like *tenebat*, *multum* or *propius* are totally uncorrelated. The Pearson[2] correlation coefficient tells us that in this sequence there is no correlation between z-score and activation-score (with a Pearson of 0.38). This example is one of the most correlated examples of our dataset, thus deep learning seems to learn more than a simple z-score.

In order to understand what the real linguistic marks found by deeplearning are (the convolution layer), we did several tests on different languages and our model seems to have the same behavior in all of them. We used a French web-platform called Hyperbase[3] to perform all the linguistic statistics tests.

[1] Titus Livius Patavinus – (64 or 59 BC - AD 12 or 17) – was a Roman historian.

[2] Pearson correlation coefficient measures the linear relationship between two datasets. It has a value between $+1$ and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative

[3] Hyperbase is an on-line (*http://hyperbase.unice.fr*) linguistic toolbox, which allows the creation of databases from textual corpus and the performing of analysis and calculations such z-score, cooccurrences, PCA, K-Means distance, ...

4.2 Dataset: English

The first dataset we used for our experiments is the well known IMDB Movie review corpus for sentiment classification. It consists of 25,000 reviews labeled by positive or negative sentiment with around 230,000 words. With the default methods given by Hyperbase, we can easily show the specific vocabulary of each class (positive/negative), according to the z-score. There are for example the words *too*, *bad*, *no* or *boring* as most indicitive of negative sentiment, and the words *and*, *performance*, *powerful* or *best* for positive. Is it enough to detect automatically if a new review is positive or not? Let's see an example excerpted from a review from December 2017 (not in the training set) on the last American blockbuster:

> *[...] **i enjoyed three moments** in the film in total , **and if i am being honest and** the person **next to me fell asleep** in the middle and started snoring during the slow space chasescenes . **the story failed to** draw me in and entertain **me the way** [...]*

In general the z-score is enough to predict the class of this kind of comment. But in this case, deeplearning seems to do better, but why? If we sum all the z-scores (for negative and positive), the positive class obtains a greater score than the negative. The words *film*, *and*, *honest* and *entertain* – with scores 5.38, 12.23, 4 and 2.4 – make this example positive. Deep learning has activated different parts of this sequence (as we show in bold/red in the exemple). If we take the sub-sequence *and if i am being honest and*, there are two occurences of *and* but the first one is followed by *if* and Hyperbase give us 0.84 for *and if* as a negative class. This is far from the 12.23 in the positive. And if we go further, we can do a co-occurrence analysis on *and if* on the training set. As we see on Figure 5, one of most specific adjectives[4] associated with *and if* is *honest*. Exactly what we found in our example.

In addition, we have the same behavior with the verb *fall*. There is *asleep* next to him. *asleep* alone is not really specific of negative review (z-score of 1.13). But with the word *fall*, *asleep* become one of the most specific (see the co-occurrences analysis - Figure 6).

[4]With Hyperbase we can focus on different part of speech.

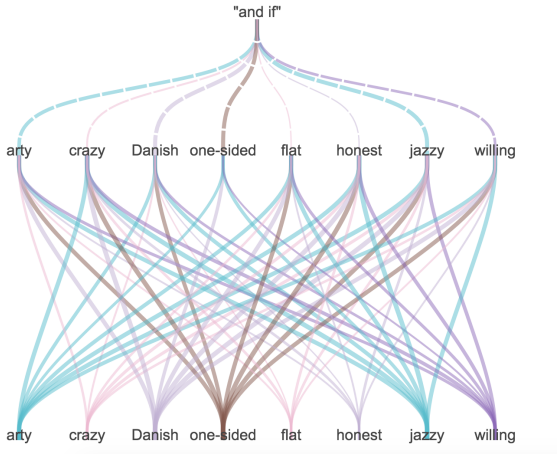

Figure 5: co-occurrences analysis of *and if* showed by Hyperbase

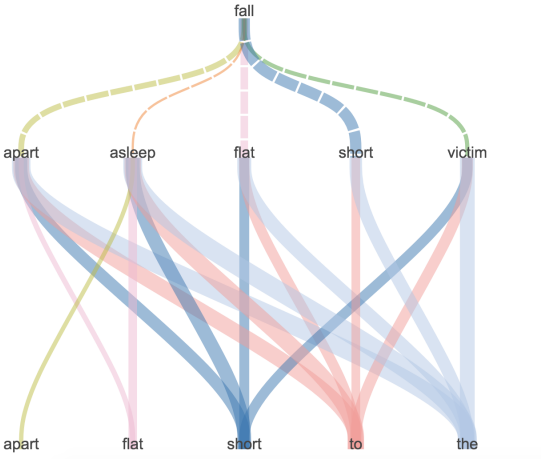

Figure 6: co-occurrences analysis of *fall* showed by Hyperbase

The activation-score here confirms that deep learning seems to focus not only on high z-score but on more complex patterns and maybe detects the lemma or the part of speech linked to each word. While the embedding is modifiable during the learning, it's possible that the final word vectors share this kind of information. We will see now that these observations are still valid for other languages and can even be generalized between different activation spikes.

4.3 Dataset: French

The French data set consists of political speeches. It's a corpus of 2.5 millions of words of French Presidents from 1958 (with C. de Gaulle, the first President of the Fifth Republic) to 2018 with the first speeches by Macron. In this corpus we re-

moved Macron's speech from the 31st of December 2017, to use it as a test data set. In this speech, the deeplearning network primarily recognizes E. Macron (the training task was to be able to predict the correct President). To achieve this task the deeplearning network seems to succeed in finding really complex patterns specific to E. Macron. For example in this sequence :

> *[...] notre pays* ***advienne à*** *l'école pour nos enfants, au travail pour l' ensemble de* ***nos concitoyens*** *pour le climat pour le quotidien de chacune et chacun d' entre vous .* ***Ces transformations profondes*** *ont commencé et se* ***poursuivront*** *avec la même force le même rythme la même intensité [...]*

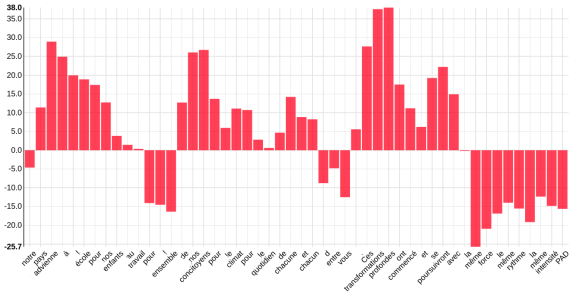

Figure 7: Deconvolution on E. Macron speech.

The z-score gives a result statistically closer to De Gaulle than to E. Macron. The error in the statistical attribution can be explained by a Gaullist phraseology and the multiplication of linguistic markers strongly indexed with de Gaulle: for example, de Gaulle had the characteristic of making long and literary sentences articulated around conjunctions of coordination as in *et* (z-score = 28 for de Gaulle, two occurrences in the excerpt). His speech was also more conceptual than average, and this resulted in an over-use of the articles defined *le*, *la*, *l´*, *les*) very numerous in the excerpt(7 occurrences); especially in the feminine singular (*la république*, *la liberté*, *la nation*, *la guerre*, etc., here we have *la même force*, *la même intensité*.

The best results given by deeplearning themselves can surprise the linguist and match perfectly with what is known about the sociolinguistics of Macron's dynamic kind of speeches.

The most important activation zone of the excerpt concerns the nominal syntagm *transformations profondes*. Taken separately, neither of the phrase's two words are very Macronian from a statistical point of view (*transformations* = 1.9 *profondes* = 2.9). Better: the syntagm itself is not attested in the President's learning corpus (0 occurrence). However, it can be seen that the co-occurrence of *transformation* and *profondes* amounts to 4.81 at Macron: so it is not the occurrence of one word alone, or the other, which is Macronian but the simultaneous appearance of both in the same window. The second and complementary activation zones of the excerpt thus concern the two verbs *advienne* and *poursuivront*. From a semantic point of view, the two verbs perfectly conspire, after the phrase *transformations profondes*, to give the necessary dynamic to a discourse that advocates change. But it is the verb tenses (borne by the morphology of the verbs) that appear to be the determining factor in the analysis. The calculation of the grammatical codes cooccurring with the word *transformations* thus indicates that the verbs in the subjunctive and the verbs in the future (and also the nouns) are the privileged codes for Macron (Figure 8).

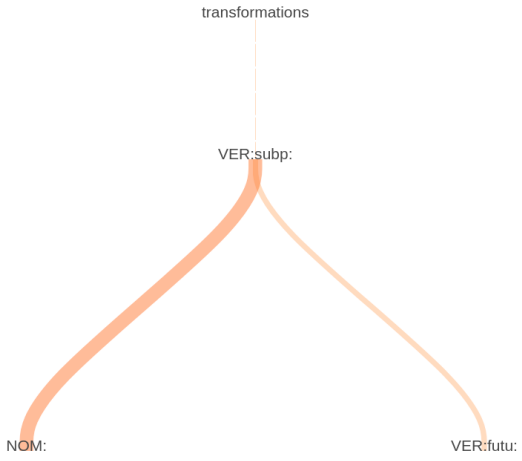

Figure 8: Main part-of-speech cooccurrences for *transformations* showed by Hyperbase

More precisely the algorithm indicates that, for Macron, when *transformation* is associated with a verb in the subjunctive (here *advienne*), then there is usually a verb in the future co-present (here *poursuivront*). *transformations profondes*, *advienne* to the subjunctive, *poursuivront* to the future: all these elements together form a speech promising action, from the mouth of a young and dynamic President. Finally, the graph indicates that *transformations* is especially associated with

nouns in the President's speeches: in an extraordinary concentration, the excerpt lists 11 (*pays*, *école*, *enfants*, *travail*, *concitoyens*, *climat*, *quotidien*, *transformations*, *force*, *rythme*, *intensité*).

4.4 Dataset: Latin

The last dataset we used is based on Latin. We assembled a contrastive corpus of 2 million words with 22 principle authors writting in classical Latin. As in the French dataset, the learning task here was to be able to predict each author according to new sequences of words. The next example is a excerpt of chapter 26 of the 23th book of Livy:

> *[...] tutus tenebat se quoad multum ac diu PAD quattuor milia peditum et quingenti equites in supplementum missi ex PAD sunt . tum refecta tandem spe* ***castra propius hostem*** *mouit classem que et ipse instrui parari que iubet ad insulas maritimam que oram tutandam . in* ***ipso impetu*** *mouendarum de [...]*

The statistics here identify this sequence with Caesar[5] but Livy is not far off. As historians, Caesar and Livy share a number of specific words: for example tool words like *se* (reflexive pronoun) or *que* (a coordinator) and prepositions like *in*, *ad*, *ex*, *of*. There are also names like *equites* (cavalry) or *castra* (fortified camp).

The attribution of the sentence to Caesar can not only rely only on z-score: *que* or *in* or *castra*, with differences thereof equivalent or inferior to Livy. On the other hand, the differences of *se*, *ex*, are greater, as is that of *equites*. Two very Caesarian terms undoubtedly make the difference *iubet* (he orders) and *milia* (thousands).

The greater score of *quattuor* (four), *castra*, *hostem* (the enemy), *impetu* (the assault) in Livy are not enough to switch the attribution to this author.

On the other hand, deeplearning activates several zones appearing at the beginning of sentences and corresponding to coherent syntactic structures (for Livy) – *Tandem reflexes spe castra propius hostem mouit* (then, hope having finally returned, he moved the camp closer to the camp of the enemy) – despite the fact that *castra* in *hostem mouit* is attested only by Tacitus[6].

There are also *in ipso metu* (in fear itself), while *in* followed by *metu* is counted one time with Caesar and one time also with Quinte-Curce[7].

More complex structures are possibly also detected by deeplearning: the structure *tum* + participates Ablative Absolute (*tum refecta*) is more characteristic of Livy (z-score 3.3 with 8 occurrences) than of Caesar (z-score 1.7 with 3 occurrences), even if it is even more specific of Tacitus (z-score 4.2 with 10 occurrences).

Finally and more likely, the co-occurrence between *castra*, *hostem* and *impetu* may have played a major role: Figure 9

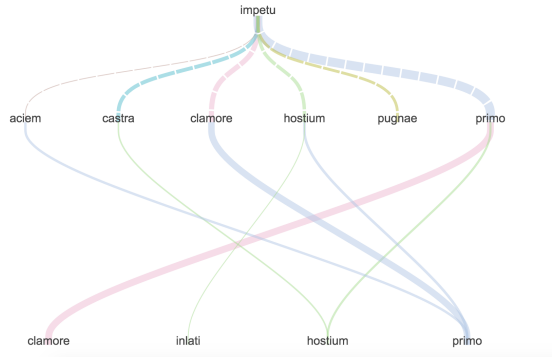

Figure 9: Specific co-occurrences between *impetu* and *castra* showed by Hyperbase.

With Livy, *impetu* appears as a co-occurrent with the lemmas *HOSTIS* (z-score 9.42) and *CASTRA* (z-score 6.75), while *HOSTIS* only has a gap of 3.41 in Caesar and that *CASTRA* does not appear in the list of co-occurrents.

For *castra*, the first co-occurent for Livy is *HOSTIS* (z-score 22.72), before *CASTRA* (z-score 10.18), *AD* (z-score 10.85), *IN* (z-score 8.21), *IMPETVS* (z-score 7.35), *QUE* (z-score 5.86)) while in Caesar, *IMPETVS* does not appear and the scores of all other lemmas are lower except *CASTRA* (z-score 15.15), *HOSTIS* (8), *AD* (10,35), *IN* (5,17), *QUE* (4.79).

Thus, all is as it should be if the deeplearning network manages to simultaniously account for specificity, phrase structure, and co-occurence networks...

[5] Gaius Julius Caesar, 100 BC - 44 BC, usually called Julius Caesar, was a Roman politician and general and a notable author of Latin prose.

[6] Publius (or Gaius) Cornelius Tacitus, 56 BC - 120 BC, was a senator and a historian of the Roman Empire.

[7] Quintus Curtius Rufus was a Roman historian, probably of the 1st century, his only known and only surviving work being "Histories of Alexander the Great"

5 Conclusion

ADT and deep learning may not be foreign continents to each other citep lebart1997. This contribution by crossing statistical approach and neural network allowed us to identify key passages and perhaps reasons that could feed our textual treatments. If the observables that presided over the detection of key passages by the ADT (the lexical specificities) are known and tested, the zones of activation of the deep learning seem to raise new linguistic observables. Recall that the linguistic matter and the topology of the passages can not return to chance: the zones of activations make it possible to obtain recognition rates of more than 90 % on the French political speech and 85 % on the corpus of the LASLA ; either rates equivalent to or higher than the rates obtained by the statistical calculation of the key passages. It remains to improve the model and to understand all the mathematical and linguistic outcomes. The first improvement that we now propose to implement is the injection of morphosyntactic information into the network in order to test ever more complex linguistic patterns.

References

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Feldman, R., and J. Sanger. 2007. *The Text Mining Handbook. Advanced Approaches in Analyzing Unstructured Data*. New York: Cambridge University Press.

L. Lebart, A. Salem and L. Berry. 1998. *Exploring Textual Data*. Ed. Springer.

S. Mellet and D. Longrée. 2009. Syntactical motifs and textual structures. In *Belgian Journal of Linguistics 23*, pages 161–173.

A Supplemental Material

In order to make our experiments reproductible, we going to detail here all the hyperparameters used in our architecture. The neural network is written in python with the library Keras (an tensorflow as backend).

The embedding use a Word2Vec implementation given by the gensim Library. Here we use the SkipGram model with a window size of 10 words and output vectors of 128 values (embedding dimension).

The textual datas are tokenized by a homemade tokensizer (witch work on English, Latin and French). The corpus is splited into 50 length sequence of words (punctuation is keeped) and each word is converted inta an uniq vectore of 128 value.

The first layer of our model takes the text sequence (as word vectors) and apply on it a weight corresponding to our WordToVec values. Those weight are still trainable during the train of the model.

The second layer is the convolution, a Conv2D in Keras with 512 filters of size $3 * 128$ (filtering three words at time), with a Relu activation method. Then, there is the Maxpooling (MaxPooling2D)

(The deconvolution model is identical until here. We replace the rest of the classification model (Dense) by a transposed convolution (Conv2DTranspose).)

The last layers of the model are a Dense layers. One hidden layer of 100 neurons with a Relu activation and one final layer of size eaqual to the number of class with a softmax activation.

All experiments in this paper share the same architecture and the same hyperparameters, and are trained with a cross-entropy method (with an Adam optimizer) with 90% of the dataset for the training data and 10% for the validation. The all the tests in this paper are done with new data not included in the original dataset.