Deconvolution for linguistic analysis

L. Vanni[1], V. Elango[2], C. Aguilar[1], D. Longrée[3], D. Mayaffre[1], F. Precioso[2], M. Ducoffe[2]
[1] Univ. Nice Sophia Antipolis - I3S, UMR UNS-CNRS 7271 06900 Sophia Antipolis, France
{lvanni, mayaffre}@unice.fr
[2] Univ. Nice Sophia Antipolis - BCL, UMR UNS-CNRS 7320 - 06357 Nice CEDEX 4, France
{ducoffe, precioso}@unice.fr - ecveer@gmail.com
[3] Univ. Liège - L.A.S.L.A, Bélgique
dominique.longree@uliege.be

Abstract

This document contains the instructions for preparing a paper submitted to COLING-2018 or accepted for publication in its proceedings. 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

Neural Networks had a tremendous impact on Natural Language Processing. They outperform the performance of many other state-of-the-art systems on a wide range of NLP tasks and are highly used in industrial technologies. Such systems rely on end-to-end training on large amounts of data, making no prior about the linguistic structure. Thus, they step away from Textual Data Analysis used in linguistic as they learn implicit linguistic information automatically. However, we do not know the richness of such information. Do neural networks make use of redundant information, also revealed with traditional textual data analysis? Or do they rely also on complementary linguistic structure, undiscovered by TDA? If so, projecting neural networks features back in the input space would highlight new linguistic structures to improve the analysis of a corpus. Our hypothesis is that deep learning is, of course, sensitive to the linguistic units on which the computation of the key statistical sentences are based, but also sensitive to other phenomena than frequency and other complex linguistic observables that the TDA has more difficult to take into account - as would be linguistic pattern (Mellet et Longrée, 2009). Our contribution confronts Textual Data Analysis and Convolutional Neural Networks for text analysis. We hijack deconvolution network for image analysis to offer to the linguistic community a new point of view for text analysis that we denote 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 over words in a sentence that promotes the patterns relevant for the classification decision. We confront z-scoring and deconvolution saliency on three languages: English, French and Latin. For all our datasets, deconvolution saliency highlights new linguistic observables, unperceivable with z-scoring.

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 on an input image, stacked layers of convolutions, nonlinearities 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 (Col- lobert and Weston, 2008). Collobert adapted CNNs for various natural language processing (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, which is concatenated as a matrix. Word embeddings are not intended to hold spatial information. Thus the width of the convolutional filters is usually the same as the dimension of the word features.

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]. Recognition task is at the heart of linguistic interests which mostly focus on contrastive analysis. Moreover, CNNs are static architectures that, according to specific tuning, 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 (2017)). Due to the convolution operators involved, they can be easily parallelized and may be also inferred on CPU, which is a practical solution to get rid of the need of GPUs at test time and widespread our tools.

All previous works converged to a common assessment: both CNNs and RNNs provide relevant, but different information for text classification. However, if 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 (). When it comes to deep architectures, Krizhevsky et al. 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. provided new visualization tools for recurrent models. They use decoders, t-SNE and first derivative saliency, in order to shed light on how neural models work. Our perspectives differ 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 exist many visualization tools for CNNs analysis, inspired by the computer vision field. Such works may help us highlighting 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 used 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, which are the two folds: i) they are computationally expensive : the first method requires to train a model for each latent representation, and the second relies on backpropagation for each submitted sentence. ii) they are highly hyperparameters' dependent and may require some tuning given the task at hand. On the other hand, Deconvolution Networks, proposed by Zeiler et al in ?, is 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 filters weights in a convolutional layer and then transposing their kernel matrix. More details of the deconvolution heuristic are provided in section **??**. Deconvolution holds several advantages. Firstly 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 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 Text Classification

We propose a deep neural model to capture linguistics partern of the text. This model is based on simple Convolutional Neural Network with an embedding layer for words representaion, one covolutional with pooling layer and finnaly one Dense layer. Figure 1 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 element (for text classification). The embedding is build on top of a Word2Vec architecture trained on a Skip-gram model. Our text tokenizer keeps all the words to make sure all linguistics material could be detected at the end by the model. This embedding is also trainable by the model to reach the best text-classification acuracy.

The Convolutional layer is based on a 2 dimensionals convolution, the same as used for pictures convolution, but with a fixed width corresponding to the max width (this size is actually equal to the embedding size). With this setting, our usage of the 2 dimensionals convolution is in reallity the same as a 1 dimensional convolution (the default convolutionnal layer for text). The only parameter we adjuste here is the height of the filter corresponding to the number of words we want to put in the filter. The goal of this approach is to be able to use the standard picture deconvolution (conv2D Transpose) methods for our model on text.

The last layer is a fully connected dense network (with one hidden layer) finishing on a output size corresponding to the number of class we attempt to train.

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 called 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 convolutional transposed layer (also called deconvolution). 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 in input a sequence of words and gives us in 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 :

$$\sum_{i=1}^{x}\sum_{j=1}^{y} a_{ij} = s_n \tag{1}$$

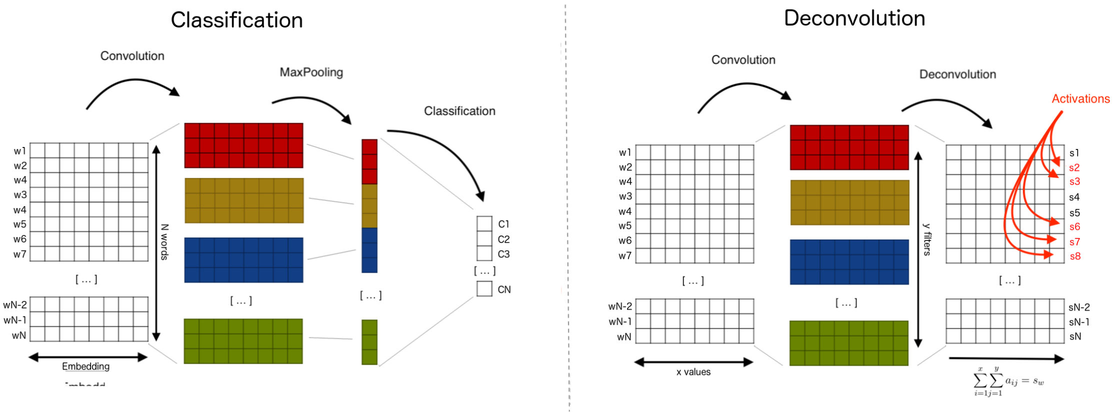

Figure 1: Deconvolution model

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 going to see now that this output of the deconvolution give us many information on how the network takes his final descision (prediction). There're well known linguistics marks encoded inside but also some more complexe pattern based on cooccurrences and maybe 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 case of a "normal" distribution. This calcul gives easily for example the most specific vocabulary of a given author in a contrastive corpora. The highest z-score are the most specific word in this case. This is a simple but strong method to analyze feature on text. It can be also used to classify word sequences according to the global z-score (sum of the score) in the sequence. The mean accuracy of this methods on our data set is around 85%, that confirm z-score is really meaningful on contrastive data. On the other hand, the deep learning reaches most of time more than 90% on text classification. It means the training methods can learn also by themselves some sort of linguistic specificities useful to distinguish class of text or authors. We've seen on image that's the role of the convolution. It learns an abstraction on the data to make classification easier. The question is : what is the nature of this abstraction on text ? We going to seen now that the deep learning detect automatically words with hight z-score but apparently it's not this only linguistic marks detected.

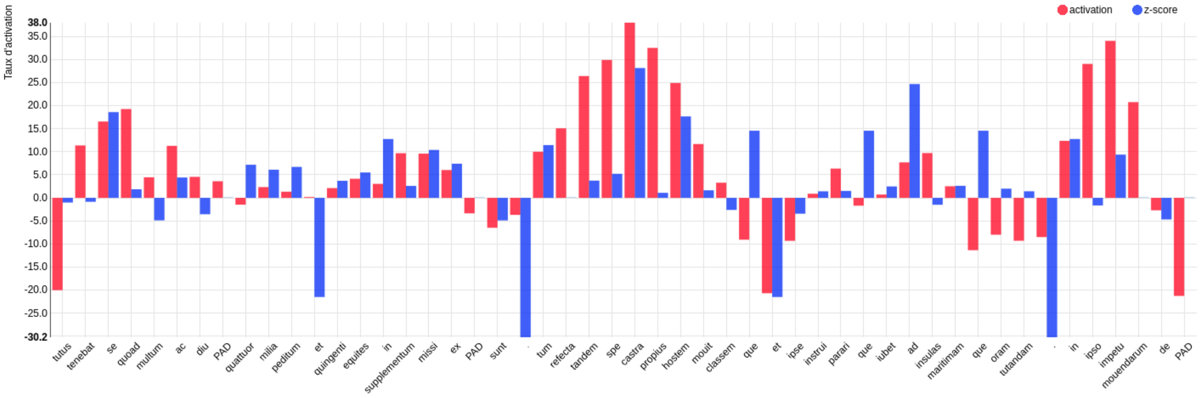

Figure 2: Latin dataset : Livy Book XXIII Chap. 26 - Z-score Vs Activation-score

The Figure 2 shows us a comparison between z-score and activation-score on a sequence extract form our latin corpora. Here it's an example where Livy[1] use some specific words. As we can see, when the z-score is the highest there are sort of activation spike around (word *castra*). But not always, for example small words as *que*, *ad* and *et* are also high in z-score but they not activate the network as the same level. We saw in (reference ****) that deeplearning is more sensible with long words, but we can see also on Figure 2 that word like *tenebat*, *multum* or *propius* are totally uncorrelated. The Pearson[2] correlation coefficient tell us on 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 example of our dataset, thus deep learning seems to learn more than a simple z-score.

In order to understand what is the real linguistic marks found by the deep learning (the convolution layer), we did several tests on different languages and our model seems to have the same behaviors on it. We use a french web platforme called Hyperbase[3] to perform all the linguistic statistics tests.

4.2 Dataset : English

The first dataset we used for our experiments is the well known IMDB Movie reviews corpus for sentiment classification. It's 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 is for example the words *too*, *bad*, *no* or *boring* as most specific of negative sentiment. And words *and*, *performance*, *powerful* or *best* for the positive. Is it enough to detect automatically if a new review is positive or not. Let's see an example extracted from a review from December 2017 (not in the training set) on the last American's 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 PAD during the slow space chasescenes .* ***the story failed to*** *draw me in and entertain* ***me the way*** *[...]*

[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, that allow to create databases from textual corpus and perform analysis and calculation on it like z-score, cooccurrences, PCA, K-Means distance, ...

In general the z-score is enough to predict the class of this kind of comment. But in this case, the deeplearning seems to do it better, why ? If we sum all the z-score (for negative and positive), positive class obtain a greater score than negative. The words *film*, *and*, *honest* and *entertain* - with scores 5.38, 12.23, 4 and 2.4 - make this example positive. The deep learning has activated different part of this sequence (As we show in bold/red in the exemple). If we take the subsequence *and if i am being honest and*, there are 2 *and* but the first one is followed by *if* and Hyperbase give us 0.84 for *and if* on negative class. It's far from the 12.23 on positive. And if we go further, we can do a cooccurrence analysis on *and if* on the training set. As we see on Figure 3, one of most specific Adjective[4] around *and if* is *honest*. Exactly what we found in our example. And around the *fall* verb, there is asleep. Here, *asleep* alone is not really specific of negative review (z-score of 1.13).

The activation-score here confirm that deep learning seems to focus not only on hight z-score but on more complex pattern and maybe detect the lemma or the part of speech linked to each word. While the embedding is trainable during the learning, it's possible that the final word vectors share this kind of informations. We going to see now that these observations are still valid on other languages and can even be generalized between different activation spike.

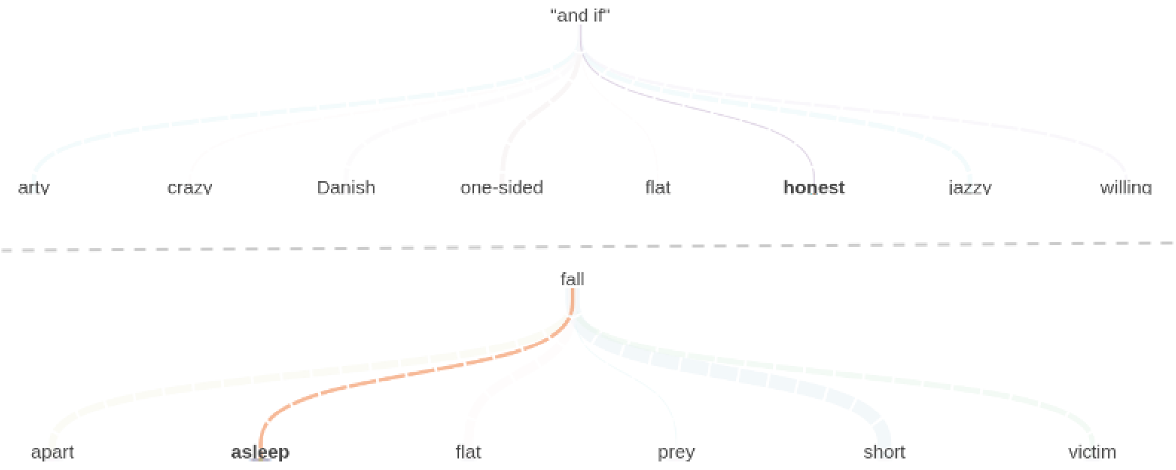

Figure 3: Main cooccurrences for *and if* and *fall* showed by Hyperbase

4.3 Dataset : French

The french data set is about political discourses. It's a corpus of 2.5 millions of words of French president from 1958 (with C. de Gaulle, the first president of the fifth republic) to 2018 with the first discourses of E. Macron. In this corpus we only remove the December 31 speech of E. Macron to use it as test data set. On this discourse E. Macron is mainly recognized by the deeplearning (The training task was to be able to predict the correct president). To achieve this task the deep learning seems to succeed to find really complex pattern 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 [...]*

The Z-score brings statistically closer to De Gaulle than E. Macron. The error of statistical attribution can be explained by a Gaullist phraseology and the multiplication of linguistic markers strongly indexed by de Gaulle: for example, de Gaulle had the characteristic of making long and literary sentences articulated around conjunctions of coordination as *et* (z-score = 28 for de Gaulle, 2 occurrences in the excerpt).

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

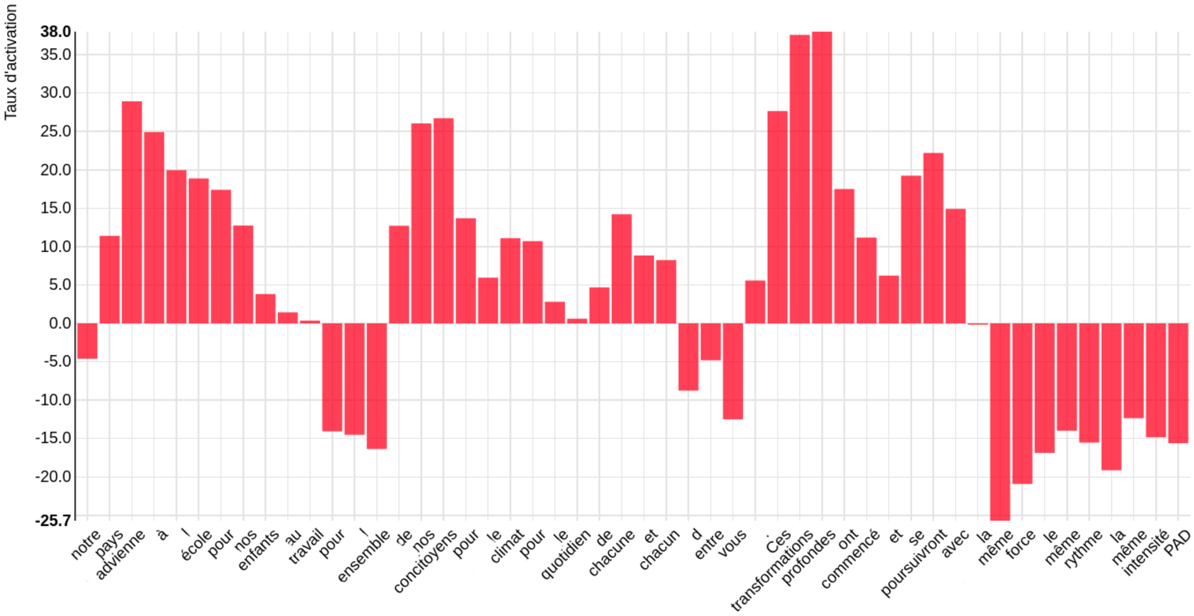

Figure 4: Deconvolution on E. Macron speech.

His speech was also more conceptual than the average, and this resulted in an over-use of the articles defined *le*, *la*, *l´*, *les*) very numerous in the extract (7 occurrences); especially in the feminine singular (*la republique*, *la liberté*, *la nation*, *la guerre*, etc., here we have *la même force*, *la même intensité*.

Les meilleures performances du deep learning interrogent quant à elle le linguiste et épouse parfaitement ce que l'on sait socio-linguistiquement du discours dynamique de Macron.

The most important activation zone of the extract concerns the nominal syntagm *transformations profondes*. Taken separately, none of the two words of the phrase are very Macronian from a statistical point of view (*transformations* = 1.9 *profondes* = 2.9). Better: the syntagm itself is not attested in the corpus of learning of the President (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 extract 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 determining in the analysis. The calculation of the grammatical codes co-occurring 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 at Macron (Figure 5).

More precisely, the algorithm indicates that, in 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 sign, together, a speech made of promise of action, in the mouth of a young and dynamic president. Finally, the graph indicates that *transformations* is especially associated with names in the President: in an extraordinary concentration, the extract 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 language. We made a contrastive corpus of 2 millions words with most of the main Latin authors (22 authors). As the French dataset, the learning task here was to be able to predict each author according to new sequence of words. The next example is a excerpt of

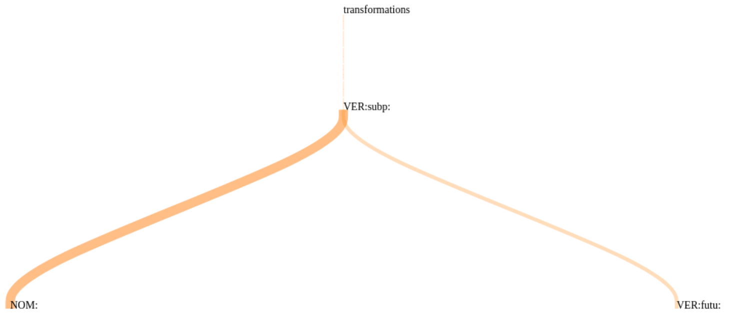

Figure 5: Main part-of-speech cooccurrences for *transformations* showed by Hyperbase

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 gives this sequence to Caesar[5] but Livy is not so far from the decision. 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 be only reliable on specificities (z-score) *que* or *in* or *castra*, with differences equivalent or inferior to Livy. On the other hand, the differences of *se*, *ex*, are superior, as 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) at Livy are not enough to switch the attribution to this author.

On the other hand, the Deep Learning activate several zones appearing at the beginning of sentence and corresponding to coherent syntactic structures (for Livy) – *Tandem reflexes spe castra propius hostem mouit* (then the hope having finally returned, it approaches 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 structure are possibly also detected by the deep learning : 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 6

impetu: - in Titus Live, appear as cooccurrents lemmas HOSTIS 9.42 and CASTRA 6.75, while HOSTIS only has a gap of 3.41 in Caesar and that CASTRA does not appear in the list of cooccurrents impetu

castra: the first cooccurent at Titus Live is HOSTIS (22,72), before CASTRA (10,18), AD (10,85), IN (8,21), IMPETVS (7,35), -QUE (5,86)) while in Caesar, IMPETVS does not appear and the scores of all other lemmas are lower except CASTRA (15,15): HOSTIS (8),, AD (10,35), IN (5,17), - THAT (4.79)

[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, author of his only known and only surviving work, "Histories of Alexander the Great"

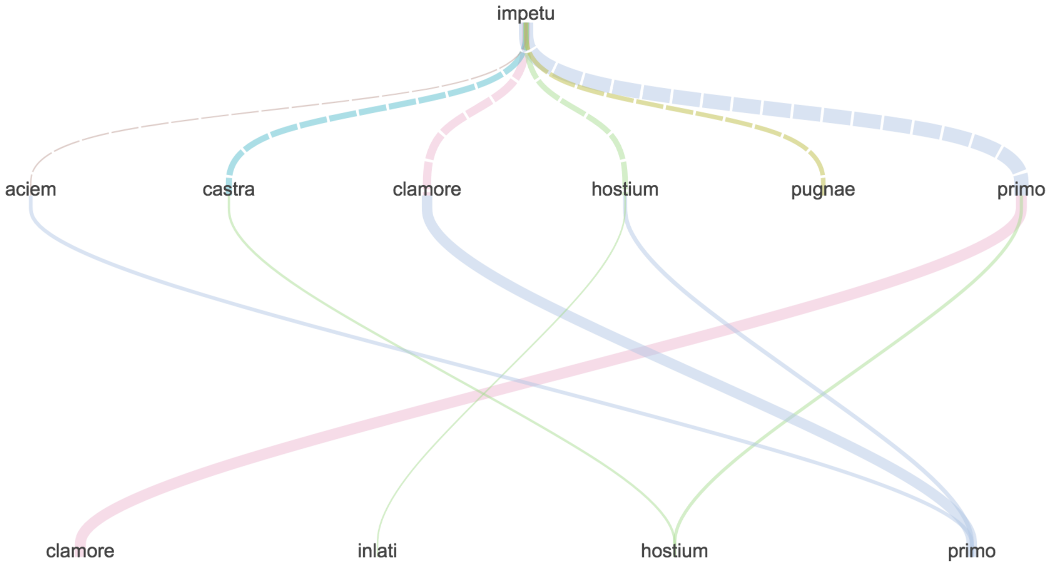

Figure 6: specific co-occurrences between *impetu* and *castra* showed by Hyperbase

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

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Association for Computing Machinery. 1983. *Computing Reviews*, 24(11):503–512.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.