Text Deconvolution Saliency (TDS): a deep tool box for linguistic analysis

Laurent Vanni[1]* Melanie Ducoffe[2]* Damon Mayaffre[1] Frederic Precioso[2]
Dominique Longrée[3] Veeresh Elango[2] Nazly Santos[2] Juan Gonzalez[2] Luis Galdo[2]
Carlos Aguilar[1]
[1]Université Côte d'Azur, CNRS, BCL, France
[2]Université Côte d'Azur, CNRS, I3S, France
[3]Univ. Liège, L.A.S.L.A, Belgium
{laurent.vanni, melanie.ducoffe}@unice.fr

Abstract

In this paper, we propose a new strategy, called Text Deconvolution Saliency (TDS), to visualize linguistic information detected by a CNN for text classification. We extend Deconvolution Networks to text in order to present a new perspective on text analysis to the linguistic community. We empirically demonstrated the efficiency of our Text Deconvolution Saliency on corpora from three different languages: English, French, and Latin. For every tested dataset, our Text Deconvolution Saliency automatically encodes complex linguistic patterns based on co-occurrences and possibly on grammatical and syntax analysis.

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) (Textual Mining)? 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. 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 Text Deconvolution Saliency (TDS). Our deconvolution saliency corresponds to the sum over the word embedding of the deconvolution projection of a given feature map. Such a score provides a heat-map of words in a sentence that highlights the pattern relevant for the classification decision. We examine z-test (see section 4.2) and TDS for three languages: English, French and Latin. For all our datasets, TDS highlights new linguistic observables, invisible with z-test alone.

2 Related work

Convolutional Neural Networks (CNNs) are widely used in the computer vision community for

* L. Vanni and M. Ducoffe contributed equally to this work and should be considered as co-first authors.

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 Kalchbrenner et al. (2014); Kim (2014). 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. 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 Yin et al. (2017).

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-test. However, such existing methods have so far encountered difficulties in underlining more challenging linguistic knowledge, which up to now have not 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 structures as in 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 the vanishing gradient problem, and thus can also model long-term dependency in a sentence Dauphin et al. (2017); Wen et al. (2017); Adel and Schütze (2017). 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 Ji and Eisenstein (2014). When it comes to deep architectures, Karpathy et al. Karpathy et al. (2015) 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. Li et al. (2015) 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 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 hidden layers 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 Zeiler and Fergus (2014), 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 3. Deconvolution has 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. (2015); 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 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. The model is trained by cross-entropy with an Adam optimizer. 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 probabilities (for text classification).

The embedding is built on top of a Word2Vec architecture, here we consider a Skip-gram model. This embedding is also finetuned by the model to to increase the accuracy. Notice that we do not use lemmatisation, as in Collobert and Weston (2008), thus the linguistic material which is automatically detected does not rely on any prior assumptions about the part of speech. In computer vision, we consider images as 2-dimensional isotropic signals. A text representation may also be considered as a matrix: each word is embedded in a feature vector and their concatenation builds a matrix. However, we cannot assume both dimensions the sequence of words and their embedding representation are isotropic. Thus the filters of CNNs for text typically differ from their counterparts designed for images. Consequently in text, the width of the filter is usually equal to the dimension of the embedding, as illustrated with the red, yellow, blue and green filters in figure 1

Using CNNs has another advantage in our context: due to the convolution operators involved, they can be easily parallelized and may also be easily used by the CPU, which is a practical solution for avoiding the use of GPUs at test time.

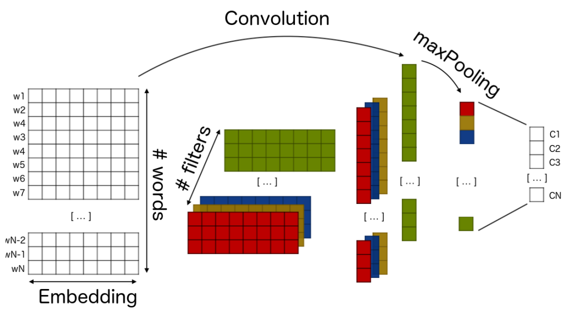

Figure 1: CNN for Text Classification

3.2 Deconvolution

Extending Deconvolution Networks for text is not straightforward. Usually, in computer vision, the deconvolution is represented by a convolution whose weights depends on the filters of the CNN: we invert the weights of the channels and the filters and then transpose each kernel matrix. When considering deconvolution for text, transposing the kernel matrices is not realistic since we are dealing with nonisotropic dimensions - the word sequences and the filter dimension. Eventually, the kernel matrix is not transposed.

Another drawback concerns the dimension of the feature map. Here feature map means the output of the convolution before applying max pooling. Its shape is actually the tuple *(# words, # filters)*. Because the filters' width (red, yellow, blue and green in fig 1) matches the embedding dimension, the feature maps cannot contain this information. To project the feature map in the embedding space, we need to convolve our feature map with the kernel matrices. To this aim, we upsample the feature map to obtain a 3-dimensional sample of size *(# words, embedding dimension, # filters)*.

To analyze the relevance of a word in a sentence, we only keep one value per word which corresponds to the sum along the embedding axis of the output of the deconvolution. We call this sum Text Deconvolution Saliency (TDS).

For the sake of consistency, we sum up our method in figure 2

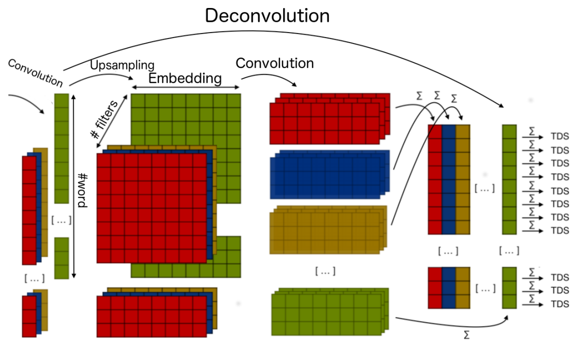

Figure 2: Textual Deconvolution Saliency (TDS)

Eventually, every word in a sentence has a unique TDS score whose value is related to the others. In the next section, we analyze the relevance of TDS. We thoroughly demonstrate empirically, that the TDS encodes complex linguistic patterns based on co-occurrences and possibly also on grammatical and syntaxic analysis.

4 Experiments

4.1 Datasets

In order to understand what the linguistic markers found by the convolutional neural network approach are, we conducted several tests on different languages and our model seems to get the same behavior in all of them. In order to perform all the linguistic statistical tests, we used our own simple linguistic toolbox Hyperbase, which allows the creation of databases from textual corpus, the analysis and the calculations such as z-test, co-occurrences, PCA, K-Means distance,... We use it to evaluate TDS against z-test scoring. We compel our analysis by only presenting cases on which z-test fail while TDS does not. Indeed TDS captures z-test, as we did not find any sentence on which z-test succeeds while TDS fails. Red words in the studied examples are the highest TDS.

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.

The second dataset targets French political discourses. It is a corpus of 2.5 millions of words of French Presidents from 1958 (with De Gaulle, the first President of the Fifth Republic) to 2018 with the first speeches by Macron. In this corpus we have removed Macron's speech from the 31st of December 2017, to use it as a test data set. The training task is to recognize each french president.

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 with the French dataset, the learning task here is to be able to predict each author according to new sequences of words. The next example is an excerpt of chapter 26 of the 23th book of Livy:

[...] tutus tenebat se quoad multum ac diu obtestanti quattuor milia peditum et quingenti equites in supplementum missi ex Africa 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 [...]*

4.2 Z-test Versus Text Deconvolution Saliency

Z-test is one of the standard metrics used in linguistic statistics, in particular to measure the occurrences of word collocations Manning and Schütze (1999). Indeed, the z-test provides a statistical score of the co-occurrence of a sequence of words to appear more frequently than any other sequence of words of the same length. This score results from the comparison between the frequency of the observerd word sequence with the frequency expected in the case of a "Normal" distribution. In the context of constrative corpus analysis, this same calculation applied to single words can readily provide, for example, the most specific vocabulary of a given author. The highest z-test are the most specific words of this given author in this case. This is a simple but strong method for analyzing features of text. It can also be used to classify word sentences according to the global z-test (sum of the scores) of all the words in the given sentence. We can thus use this global z-test as a very simple metric for authorship classification. The resulting authorship of a given sentence is for instance given by the author corresponding to the highest global z-test on that sentence compared to all other global z-test obtained by summing up the z-test of each word of the same sentence but with the vocabulary specificity of another author. The mean accuracy of assigning the right author to the right sentence, in our data set, is around 87%, which confirms that z-test is indeed meaningful for

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

Table 1: Test accuray with z-test and Deep Learning

contrast pattern analysis. On the other hand, most of the time CNN reaches an accuracy greater than 90% for text classification (as shown in Table 1).

This means that the CNN approaches can learn also on their own some of the linguistic specificities useful in discriminating text categories. Previous works on image classification have highlighted the key role of convolutional layers which learn different level of abstractions of the data to make classification easier.

The question is: what is the nature of the abstraction on text?

We show in this article that CNN approach detects automatically words with high z-test but obviously this is not the only linguistic structure detected.

To make the two values comparable, we normalize them. The values can be either positive or negative. And we distinguish between two thresholds[1] for the z-test: over 2 a word is considered as specific and over 5 it is strongly specific (and the oposite with negative values). For the TDS it is just a matter of activation strength.

The Figure 3 shows us a comparison between z-test and TDS on a sentence extracted from our Latin corpora (Livy Book XXIII Chap. 26). This sentence is an example of specific words used by Livy[2]. As we can see, when the z-test is the highest, the TDS is also the highest and the TDS values are high also for the neighbor words (for example around the word *castra*). However, this is not always the case: for example small words as *que* or *et* are also high in z-test but they do not impact the network at the same level. We can see also on Figure 3 that words like *tenebat*, *multum* or *propius* are totally uncorrelated. The Pearson cor-

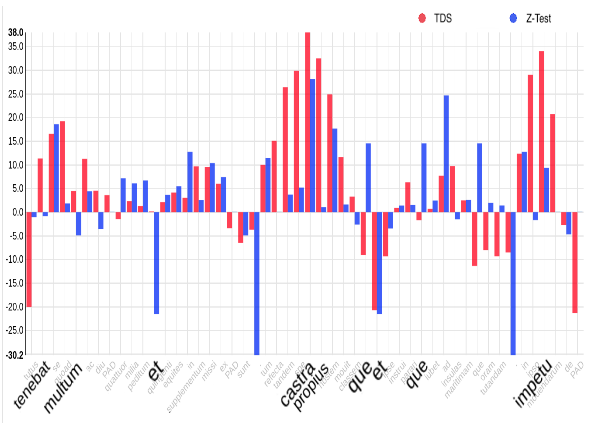

Figure 3: z-test versus Text Deconvolution Saliency (TDS) - Example on Livy Book XXIII Chap. 26

relation coefficient[3] tells us that in this sentence there is no linear correlation between z-test and TDS (with a Pearson of 0.38). This example is one of the most correlated examples of our dataset, thus CNN seems to learn more than a simple z-test.

4.3 Dataset: English

For English, we used the IMDB movie review corpus for sentiment classification. With the default methods, we can easily show the specific vocabulary of each class (positive/negative), according to the z-test. 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-test is sufficient to predict the class of this kind of comment. But in this case, the CNN seems to do better, but why?

[1]The z-test can be approximated by a normal distribution. The score we obtain by the z-test is the standard deviation. A low standard deviation indicates that the data points tend to be close to the mean (the expected value). Over 2 this score means there is less than 2% of chance to have this distribution. Over 5 it's less than 0.1%.

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

[3]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

If we sum all the z-test (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. CNN has activated different parts of this sentence (as we show in bold/red in the example). 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 our toolbox gives 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 with our co-occurrence analysis[4] (Figure 4), *honest* is among the most specific adjectivals[5] associated with *and if*. Exactly what we found in our example.

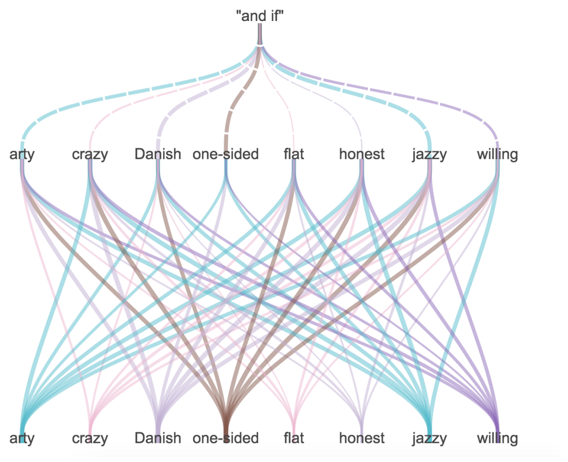

Figure 4: co-occurrences analysis of *and if* (Hyperbase)

In addition, we have the same behavior with the verb *fall*. There is the word *asleep* next to it. *Asleep* alone is not really specific of negative review (z-test of 1.13). But the association of both words become highly specific of negative sentences (see the co-occurrences analysis - Figure 5).

[4]Those figures shows the major co-occurrences for a given word (or lemma or PartOfSpeech). There two layers of co-occurrences, the first one (on top) show the direct co-occurrence and the second (on bottom) show a second level of co-occurrence. This level is given by the context of two words (taken together). The colors and the dotted lines are only used to make it more readable (dotted lines are used for the first level). The width of each line is related to the z-test score (more the z-test is big, more the line is wide).

[5]With our toolbox, we can focus on different part of speech.

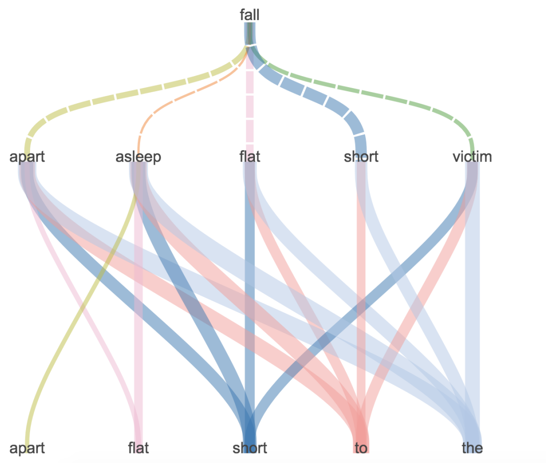

Figure 5: co-occurrences analysis of *fall* (Hyperbase)

The Text Deconvolution Saliency here confirms that the CNN seems to focus not only on high z-test but on more complex patterns and maybe detects the lemma or the part of speech linked to each word. We will see now that these observations are still valid for other languages and can even be generalized between different TDS.

4.4 Dataset: French

In this corpus we have removed Macron's speech from the 31st of December 2017, to use it as a test data set. In this speech, the CNN primarily recognizes Macron (the training task was to be able to predict the correct President). To achieve this task the CNN seems to succeed in finding really complex patterns specific to 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-test gives a result statistically closer to De Gaulle than to 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: De Gaulle had the specificity of making long and literary sentences articulated around co-ordination conjunctions as in *et* (z-test = 28 for de Gaulle, two oc-

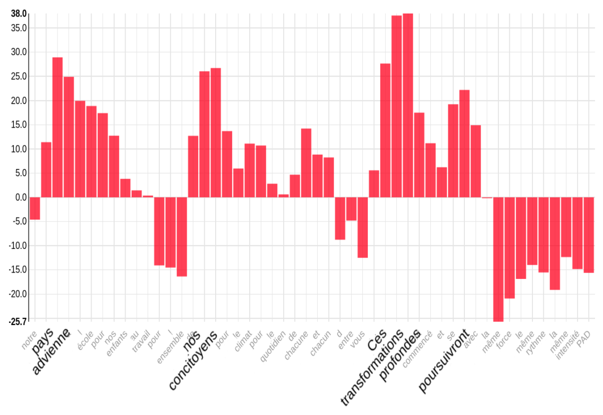

Figure 6: Deconvolution on Macron speech.

currences 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 the CNN may be surprising for a linguist but match perfectly with what is known about the sociolinguistics of Macron's dynamic kind of speeches.

The part of the excerpt, which impacts most the CNN classification, is related to 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 does not appear 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 most impacting part of the excerpt thus is related to the two verbs *advienne* and *poursuivront*. From a semantic point of view, the two verbs perfectly contribute, after the phrase *transformations profondes*, to give the necessary dynamic to a discourse that advocates change. However it is the verb tenses (carried by the morphology of the verbs) that appear to be the determining factor 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 for Macron (Figure 7).

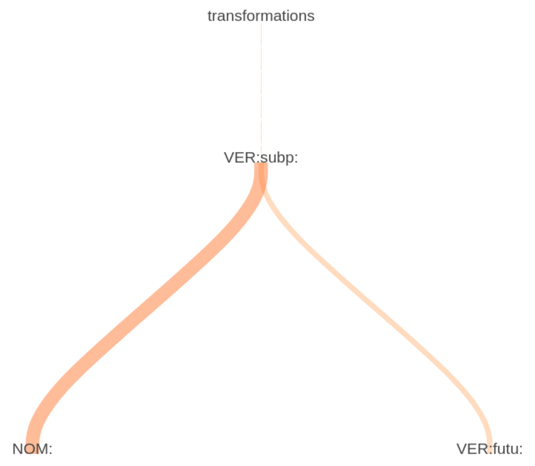

Figure 7: Main part-of-speech co-occurrences for *transformations* (Hyperbase)

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*).

4.5 Dataset: Latin

As with the French dataset, the learning task here is to be able to predict the identity of each author from a contrastive corpus of 2 million words with 22 principle authors writting in classical Latin.

The statistics here identify this sentence as Caesar[6] 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 nouns like *equites* (cavalry) or *castra* (fortified camp).

The attribution of the sentence to Caesar cannot only rely only on z-test: *que* or *in* or *castra*, with

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

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, CNN 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[7].

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[8].

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

Finally and more likely, the co-occurrence between *castra*, *hostem* and *impetu* may have played a major role: Figure 8

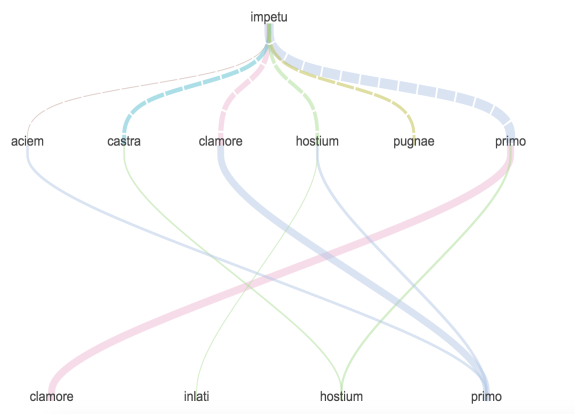

Figure 8: Specific co-occurrences between *impetu* and *castra* (Hyperbase)

With Livy, *impetu* appears as a co-occurrent with the lemmas *hostis* (z-test 9.42) and *castra* (z-test 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-test 22.72), before *castra* (z-test 10.18), *ad* (z-test 10.85), *in* (z-test 8.21), *impetus* (z-test 7.35), *que* (z-test 5.86) while in Caesar, *impetus* does not appear and the scores of all other lemmas are lower except *castra* (z-test 15.15), *hostis* (8), *ad* (10,35), *in* (5,17), *que* (4.79).

Thus, our results suggest that CNNs manage to account for specificity, phrase structure, and co-occurence networks...

4.6 Preprocessings and hyperparameters

In order to make our experiments reproductible, we 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 uses 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 home-made tokensizer (which 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 unique vector of 128 value.

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

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 Dense layers. One hidden layer of 100 neurons with a Relu activation and one final layer of size equal to the number of classes with a softmax activation.

All experiments in this paper share the same architecture and the same hyperparameters, and

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

[8]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"

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. All the tests in this paper are done with new data not included in the original dataset.

5 Conclusion

In a nutshell, Text Deconvolution Saliency is efficient on a wide range of corpora. By crossing statistical approaches with neural networks, we propose a new strategy for automatically detecting complex linguistic observables, which up to now hardly detectable by frequency-based methods. Recall that the linguistic matter and the topology recovered by our TDS cannot return to chance: the zones of activation make it possible to obtain recognition rates of more than 91% on the French political speech and 93% on the Latin corpus; both rates equivalent to or higher than the rates obtained by the statistical calculation of the key passages. Improving the model and understanding all the mathematical and linguistic outcomes remains an import goal. In future work, we intend to thoroughly study the impact of TDS given morphosyntactic information.

Acknowledgments

This work has been partly funded by the French Government (National Research Agency, ANR) through the grant ANR-16-CE23-0006 *Deep in France*, through the "Investments for the Future" Program ANR-11-LABX-0031-01, and through the UCAJEDI Investments in the Future project ANR-15-IDEX-01.

References

Heike Adel and Hinrich Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1723–1729.

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.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pages 933–941.

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

Hyperbase. Web based toolbox for linguistics analysis. http://hyperbase.unice.fr.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 13–24.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 655–665.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

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

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

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

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 438–449.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.