IRISA Rennes

Research Project- Applied Mathematics - INSA Rennes

A statistical - geometrical study of the attention based models in natural language processing

Student :
Loïc FOSSE

School Year :
2021 / 2022

Professional Supervisor :
Guillaume Gravier
Duc-Hau Nguyen

Academic Supervisor :
Abdelaziz Belmiloudi

Acknowledgement :

I would like to thank Mr Gravier, director of the IRISA laboratory, for having allowed me to reintegrate the Linkmedia [1] team of the IRISA laboratory.

I was able to work for more than a year in the field of statistics research with Duc-Hau Nguyen, a PhD student at IRISA, whom I would like to thank also for helping me in my different projects. These experiences were very enriching.

I learned a lot through the different discussions I had with Mr Gravier who knew how to direct my different projects in a remarkable way. I also learned a lot by the side of Duc-Hau Nguyen, who taught me all the informatic tools that I needed for my research, but also all the theory behind language processing.

I am now more than sure that I want to continue in the field of research in applied mathematics.

[1] http://www.irisa.fr/equipes/linkmedia

Introduction :

Il paraît raisonable d'énoncer aujourd'hui que le domaine du traitement des langues fait parti intégrante de nos vies. Il est donc aujourd'hui primordial de comprendre les modèles sous-jacents. Après l'arrivée des architectures *transformers* [Vaswani et al., 2017] en 2017 dans le domaine du traitement des langues et l'utilisation en masse du mécanisme d'attention, un véritable débat s'est crée dans le communauté scientifique. Certains pensent que ce mécanisme permet de rendre les modèles expliquables et de comprendre comment les modèles prennent leurs décision dans des tâches complexes, pendant que d'autres pensent que ce mécanisme n'explique en rien les décisions du modèle. Nous décidons ici d'étudier de manière mathématiques (*via* la géométrie) le comportement du mécanisme d'attention. Nous montrerons que le caractère interprétable du mécanisme d'attention réside plus dans la géométrie des plongements formés par ces modèles et que l'attention ne semble être finalement être que moyen pour construire des "cônes de classe" dans des tâches de classification, ne permettant ainsi pas de faire une explication directe.

Une question demeurera : comment utiliser cette géométrie pour comprendre le mécanisme de décision du modèle ?

Abstract :

It seems reasonable to state today that the field of language processing is an integral part of our lives. It is therefore essential to understand the underlying models. After the arrival of the *transformers* [Vaswani et al., 2017] in 2017 in the field of language processing and the mass use of the attention mechanism a real debate has been created in the scientific community. Some think that this mechanism allows to make models explainable and to understand how models make decisions in complex tasks, while others think that this others think that this mechanism does not explain the model's decisions at all. We decide here to study in a mathematical way (through the geometry) the behavior of the attention mechanism. We will show that the interpretability of the attention mechanism lies more in the geometry of the embeddings formed by these models and that attention seems to be only a means to build "class cones" in classification tasks, thus not allowing to make a direct explanation.

One question remains: how can we use this geometry to understand the decision mechanism of the model?

Contents

0.1 Introduction and first notions

In order to avoid any misunderstanding of notations and main concepts, this first section will aim at introducing the basic concepts of the language processing world.

Natural language processing is a branch of computer science which is very broad, very dense and which allows to manipulate many mathematical concepts (graph theory, statistics, optimization, . . .). During this project we limit ourselves to the processing of natural languages via statistical learning models (i.e. deep learning). In this branch, the two major concepts to be mastered are the notion of **tokens** and the notion of word embeddings.

0.1.1 The notion of tokens

In language processing for algorithmic considerations, it is necessary to consider a sentence not as a block, but as a list of blocks that we call tokens.

In natural language, sentences are composed of bricks called words. The tokens are the elementary bricks that constitute sentences but in the context of language processing. The simplest way to "tokenize" a sentence is to consider a word = a token. However in the present context, this way of proceeding is (almost) never used. The way to build tokens is very complex and requires advanced statistical methods like *WordPiece* [Wu et al., 2016] or *SubWord* [Bojanowski et al., 2017]. To simplify, to a word we can make correspond several tokens. For example if a word is a verb, we can split this verb in two tokens : stem + ending. The reality of things is more complex, but for the purposes of this report, keeping this example in mind is sufficient.

Thus, in an sentence s, the sentence is cut up word by word, then each word is cut up again according to a very precise algorithm, in order to finally obtain the tokens. Figure 1 gives an example of the tokens obtained on an artificial sentence.

It is important to remember that tokens, are the elementary building blocks for constructing sentences in language processing, and that now a sentence will no longer be seen as a succession of words but as a list (in the computer sense) of tokens.

sentence	example of a tokenized sentence
tokens	[CLS] example of a token ##ized sentence [SEP]
ids	101 2742 1997 1037 19204 3550 6251 102

Figure 1: example of a tokenized sentence with the SubWord [Bojanowski et al., 2017]

Thus from now on, a sentence s will be a sequence of tokens. $T(s)$ will represent the number of tokens in a sentence, and for every $i \in \{0, ..., T(s) - 1\}$, $s[i]$ will denote the token at the position i of the sentence s.

0.1.2 The notion of word embeddings

The notion of tokens being introduced, we can introduce the notion word embedding. In language processing, the different tokens pass through statistical models for prediction purposes (or other). However, today's stats models, only accept numerical data as input (numbers, vectors, matrices, ...), but the *tokens* are still textual data, given the previous definition. We will call the continuous high dimensional vector representation of a *token* the embedding of a token. If we consider a sentence s and $T(s)$ the number of *tokens* in this sentence then the embedding of the i^{th} *token* will be $e_i(s) \in \mathbb{R}^d$ a real vector of dimension d[2]. For an entire sentence s we will call the embedding matrix $W(s) \in \mathbb{R}^{T(s)\times d}$ the matrix which will contain row by row the different embeddings of the tokens.

These different embeddings are not obtained randomly because they must respect some important properties (words close in meaning must have "close" vector representations) and nowadays several very efficient algorithms allow to obtain word embeddings with very interesting properties. These algorithms are called *Skip Gram* and *CBOW* [Xiong et al., 2019, Mikolov et al., 2013]. The embeddings obtained from these algorithms are called non-contextual embeddings, because their construction does not take into account the context of the whole sentence.

The main statistical models in language processing work on the embeddings of the tokens of a given sentence. Like most statistical models, they are organized in layers each layer representing a sequence of mathematical operations. Generally speaking, each layer will consider an embedding matrix $W_{in}(s) \in \mathbb{R}^{T(s)\times d_{in}}$ as input and, through several operations, will give as output a new matrix $W_{out}(s) \in \mathbb{R}^{T(s)\times d_{out}}$ which will be used to feed the next layer. A model thus simply aims at transforming the embeddings of a given sentence, through several layers. This new version of the embeddings obtained at the output of each layer will be contextualized because the construction of the embeddings takes into account the whole sentence.

0.1.3 The notion of vocabulary

A difficulty in language processing is that each sentence is composed of a number of tokens, which is different, and that each sentence will be composed of different tokens. These difficulties seem obvious when stated, but as is often the case in mathematics/computer science, what seems obvious poses the most problems. How can we make sure that each sentence is considered in the same way as the input of

[2] in the following we will handle embeddings where $d = 300$

the learning model, and that each sentence can be passed into the model? How does the model associate each word with its embedding in an efficient way and then create in an efficient way the embedding matrix of a sentence ?

The problem of size between the different sentences can be solved in a fairly simple way by adding tokens called padding. For the second problem which is the difference of tokens between each sentence and finally the association of each tokens with its initial padding, we have to introduce the notion of vocabulary. The vocabulary is finally an associative table (Hash Map or dictionary), which will allow for each unique tokens among all the possible *tokens* to match an integer which will be its identifier. The table will finally have the form {tokens : id}. The identifier is the "position" of the *tokens*, in the dictionary.

Its meaning can vary according to the models, but the most common case is the following if we consider T the total number of unique *tokens*, and if we consider $W \in \mathbb{R}^{T \times d}$ the matrix gathering all the non-contextual plungings of the *tokens*, then the identifier of each *tokens* corresponds to its row in the matrix W.

The way each sentence s is treated by the model is as follows:

- transformation of the sentence into tokens (with addition of the padding tokens)
- transformation of the tokens into identifiers (passage in the vocabulary)
- construction of the non-contextual embedding (construction of the matrix $W(s) \in \mathbb{R}^{T(s) \times d}$)
- entry into the model and progressive construction of the contextual embeddings
- final task (classification, regression, . . .)

In the framework of the models that we will use during this report, the construction of the non-contextual embeddings is a layer of the considered model (it will correspond to the 0 layer / Embedding layer). The construction of the non-contextual embedding will be realized via a complex algorithm and the parameters of this algorithm will be modified during the training of the model.

This is not always the case. In many models the non-contextual layer is frozen, *i.e.* the non-contextual embeddings are built before the model and during the training of the model are not modified. They are considered as constant, input variables of the model.

0.1.4 Summary and transition

Through this introductory section, we have been able to introduce the notions of tokens and word embeddings and how we associate the two through vocabulary. It is now time to move on to the analysis of the heart of the report.

The presentation of the results will be divided in the following way:

- presentation of the datasets and architectures used during the study
- presentation of the results and the contributions in relation to the state of the art
- conclusion on the results and the perspectives to improve the study

For the purpose of reproducibility of the experiments, a git[3] has been made available

[3] https://github.com/Kihansi95/ExplanationPairSentencesTasks

0.2 Presentation of the data

We will consider here classification tasks with different datasets. We will deal with three different datasets in order to illustrate our results on different data and thus make our results more consistent.

0.2.1 Inference in Natural Language (INL)

The INL task is a very famous and difficult task. In this task we are given two sentences the premise and the hypothesis. The objective is to predict the link between these two sentences among: entailment, neutral, contradiction.

- Neutral (0) means that there is no logical link between the two sentences
- Entailment (1) means that the hypothesis is the logical following of the premise.
- Contradiction (2) means the opposite

The *SNLI* dataset [Bowman et al., 2015] is a famous one in NLP and will be the one we'll use. But this dataset is not enough, we'll need for this study an annotation on this dataset *i.e.* for each sentence we'll need a mark on each token which justify the label. This means that for each sentence s we need a binary vector α such that $\alpha(i) = 1$ is equivalent to the fact that the token at the position i in the sentence s is important to justify the label of the sentence.

The *E-SNLI* [Camburu et al., 2018] dataset respects this criterion and is an augmented version of the previous dataset and will be the version of the dataset we will use. The figure 2 gives us a good view of the dataset and the annotation. This annotation was made by humans and according to a very precise guide. It is therefore subject to bias and criticism.

Note that, the fact that there are two sentences will not change our doing things. The two sentences will be concatenated, and this will be treated as a single sentence.

0.2.2 Hatexplain

Detecting hate speech is unfortunately a major issue in today's world, especially on social networks like tweeter. Hate speech is filled with bias, especially with the presence of slurs and certain word groups/families. This is the reason why we'll use the *HateXplain* [Mathew et al., 2021] from the Huggin Face. The objective of this dataset is to see the behavior of the statistical models in line with these different biases. This dataset is more classical, with only one sentence for each label. In this dataset we will deal also with three classes:

- Normal (0) means that there is nothing particular in the sentence.
- HateSpeech (1)
- Offensive (2)

We can note that the last two classes are really close which will make the classification task more difficult. Like the *E-SNLI* dataset this dataset comes with an annotation also made by humans following and precise guide which provide us an explanation on the class of each sentence.

0.2.3 YelpHat

The human attention map data set or YelpHat, [Sen et al., 2020] is the last dataset we will use in this study. This dataset was made to answer the question: does the humans and the attention mechanism (which we will introduce later) focus on the same words for the text classification. This task is much easier than the previous one. This dataset contains reviews of restaurant and there are only two classes:

- bad (0)
- good (1)

The objective of the class is two identify the polarity of the sentence, which is a very classical task in NLP. With this dataset also comes an annotation. This annotation is once again made by humans and gives us the important, tokens for the classification task.

Now that we have introduced the basic language processing data and tools, we can move on to the heart of this report.

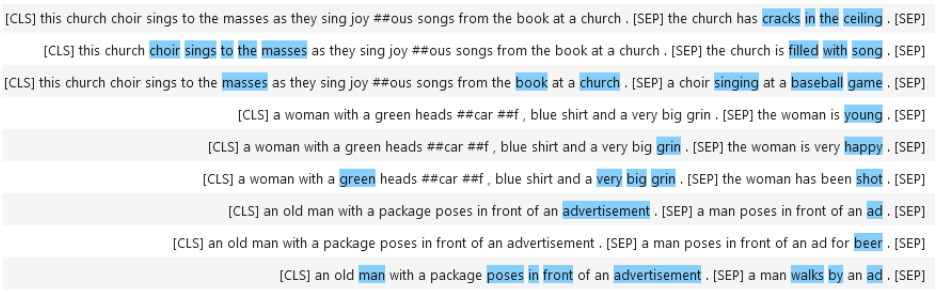


Figure 2: Annotation on the E-SNLI set. For the second sentence the corresponding class is entailment, according to the human annotation only the tokens: choir, sings, to, the, masses, filled, with, songs, allow to determine this class. The first sentence is part of the class neutral this class is special because in the annotation guide only the words on the second sentence can be underlined as shown in this example.

0.3 Around the attention model

The context of the study is as follows. Following previous studies on BERT-type [Devlin et al., 2019] networks in classification tasks, we have observed interesting behaviors in the so-called attention mechanism. This mechanism is extremely complex, especially in BERT-type architectures. We decided here to take the attention mechanism in its simplest version, to reconstruct a simple model based on the attention mechanism in order to better understand the behaviors we observed in more complex networks.

0.3.1 The attention mechanism

The attention mechanism is now famous and allowed to reach high-performance results in many NLP tasks. There are many forms to express the attention of a given sentences s. Here we will study the auto-attention scaled dot product [Vaswani et al., 2017]. Thus given a sentence s and its embedding matrix $W(s) \in \mathbb{R}^{T(s)\times d}$, we will consider three projections of this matrix into, three different spaces: the key, the value and the query spaces. The projections are only the results of matrix products :

- $Q_W(s) = W(s) \times Q$ will be the projections on the query space.
- $K_W(s) = W(s) \times K$ will be the projections on the key space.
- $V_W(s) = W(s) \times V$ will be the projections on the value space.

where $Q, K, V \in \mathbb{R}^{d\times d}$ are the weights of our model and will be learn by classical gradient descent. The auto-attention matrix will be define as :

$$A(W(s)) := softmax\left(\frac{Q_W(s)K_W(s)^T}{\sqrt{d}}\right) \in \mathbb{R}^{d\times d} \tag{1}$$

We can thus obtain a new representation of the embeddings W by computing:

$$W_{out}(s) = A(W(s)) \times V_W(s) \quad \in \mathbb{R}^{T(s)\times d}$$

These equations define a single head **attention layer**. We propose here a study of this mechanism in classification tasks. The objective is to understand better the behavior of the embeddings and their projections on to different spaces, in a simple architecture.

0.3.2 The architecture for text classification

The architecture we will use for classification is a classic one and will be based on the $[CLS]$ token.

At the beginning of each sentence s we'll put the special token $[CLS]$. This new sentence will go through the architecture based on attention. Finally the embedding of the $[CLS]$ token (*i.e.* $W_{out}(s)(0,.) \in \mathbb{R}^d$ since it's the first token of the sentence) will be used to classify the sentence through a simple feed forward layer. Figure 3 sums up the architecture we used for our study.

First of all here the embedding layer will be initiated with the well known GloVe [Pennington et al., 2014] embeddings of dimension $d = 300$.

In this architecture, the *positional encoding* [Vaswani et al., 2017] is added initially to our embedding. This positional embedding for every sentence s, will allow each embedding to encode the position of each token in the sentence.

After each attention layer a *normalization layer*, is added to help during the training part, [Ba et al., 2016]. The normalization layer will add the input and the output of the attention layer and normalize the sum. This layer is here to prevent from vanishing gradient problem, which is well known in attention type models.

Thus, in these models, only the $< cls >$ token we'll be used to proceed the classification. The motivation of using a $< cls >$ token for the classification task, is to have a neutral token for each sentence, which will learn the context of the sentence through the model (it helps to lower the bayes between the sentences).

This model is the model for only one attention block. It is possible to add attention blocks over attention blocks .... If such things are done, we will denote by the index l the different quantities.

- $W_l(s)$, will be the embedding matrix obtained after the l^{th} attention block. If L is the number of *attention blocks* we have then $W_{out}(s) = W_L(s)$. $W_0(s)$ will corresponds to the initial embeddings, after the positional encoding.

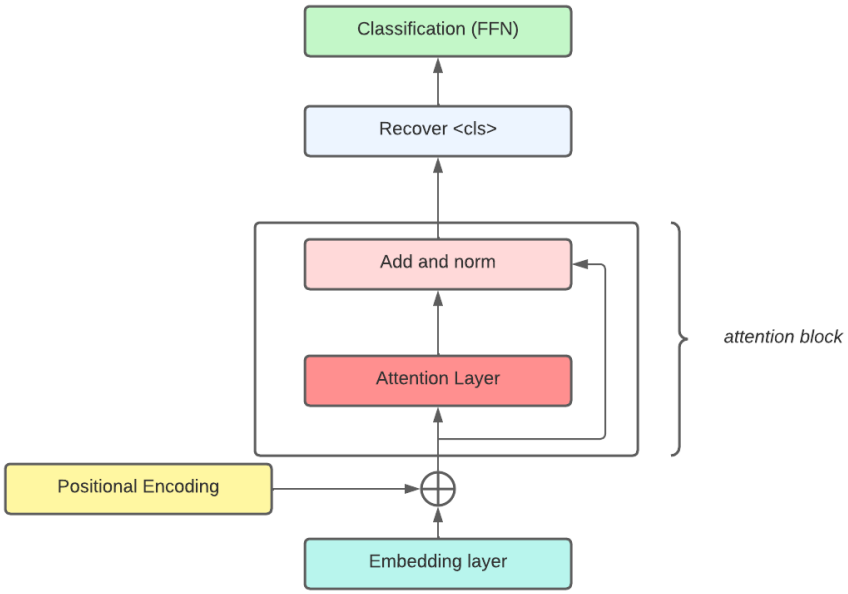


Figure 3: Architecture of the pure attention model.

- On this same basis, we will have $K_{W_{l-1}}(s)$, $Q_{W_{l-1}}(s)$ and $V_{W_{l-1}}(s)$ respectively, the key, query and value embeddings, obtained at the layer l (not $l-1$) of our model.

0.4 Methodology

0.4.1 Interpretation of the attention weights

Like we said in Section 0.3.2, the classification will only be based on the $[CLS]$ token. Thus, the behavior of the embedding of this token will be important, to understand how the model proceed the classification. If we focus on the model with only one *attention block*, thanks Equation 1, the embedding of the $< cls >$ token will have (before normalization) the following expression:

$$\begin{aligned} \{W_{out}(s)(0,k)\}_{k\in\{0,\dots,T(s)-1\}} &= \{[A(W(s)) \times V_W(s)](0,k)\}_{k\in\{0,\dots,T(s)-1\}} \\ &= \{\sum_{i=1}^{T(s)}[A(W(s))](0,i) \times [V_W(s)](i,k)\}_{k\in\{0,\dots,T(s)-1\}} \end{aligned} \tag{2}$$

Equation 2 gives us the expression of the component $k \in \{0,\dots,T(s)-1\}$ of the vector $W_{out}(0,.)$.

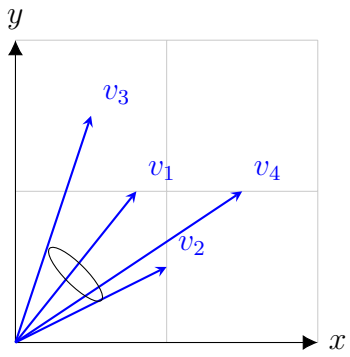


Figure 4: A quick interpretation of Equation 2 in two dimension. Here the cone of the value vectors is represented, the classification token will lie in it.

Since $A(W(s)) \geq 0$ and $\sum_{i=0}^{T(s)-1} A(W(s))(0,i) = 1$, before the normalization the embedding of the classification token will only be a convex combination of the value embeddings [4]. Thus, the embedding of this token will be within the cone shaped by the embedding values, as represented on Figure 4.

Based on Equation 2 we understand the necessity of understanding the behavior of the vector $A_W(0,.)$ and the geometry of the embedding in the different spaces.

0.4.2 The study of the geometry

On the first hand, studies like [Ethayarajh, 2019] and [Fosse et al., 2022] tend to show that the embeddings provided by the attention mechanism will lie in a narrow cone. Our objective is to push far more this notion of narrow cone.

First of all, we provide here a metric which will help quantify how narrow the cone shaped by the row of a matrix $M \in \mathbb{R}^{T(s)\times d}$ is:

$$SIM(M) = \frac{2}{T(s)(T(s)-1)} \sum_{i<j} \left[(MM^t). \frac{1}{\sqrt{diag(MM^T)diag(MM^T)^T}} \right] (i,j) \quad \in [-1,1] \tag{3}$$

It's easy to see that this metric takes its values between -1 and 1 since it is only the sum of all the pairwise cosines between the row of the matrix and noramlized by the number of pairwise possible. For the interpretation $SIM(M) = 1$ is similar to having all the row of the matrix colinear and pointing

[4] projection of the original embedding on the value space

to the same direction, $SIM(M) = 0$ is like having orthogonal rows, and finally, $SIM(M) = -1$ is like having aligned rows but pointing to opposite directions.

We will use this metric to understand the geometric properties of the different matrices K_W, V_W. Our hypothesis is that the the row of these matrix will also shape a narrow cone, and more than that the cone shaped by these rows isn't only a property of the sentence but a property of the class as explained in [Fosse et al., 2022]. All the sentences which belong to the same class, will see their projections go into the same cone. It is believed that the quality of the projections in these cones and separability of this cone is strongly related to the performance of these models.

0.4.3 The study of the attention weights

On the other hand, the study of the attention weights (i.e the vector $A(W(s))(0, .)$) is currently the center of many research subjects and creates a dense debate in the NLP community [Bibal et al., 2022]. Some, believe that the attention weights can provide an explanation on how the model makes its decision for classification and how it builds it.

Thanks to Equation 2 we can easily understand that the direction of the classification token will be strongly carried by the value embeddings $[V_W(s)](i, .)$ for the one which $[A(W(s))](0, i)$ is high.

Then the tokens at the position i for which $[A(W(s))](0, i)$ is high, are the principal tokens, and are the ones which participate the most, to the construction of the classification token and consequently to the classification.

The ground truth shows that, in general, we have all the values $A(W)(0, i)$ really similar. In other terms, the Shannon entropy [Shannon, 1948] of the vector $[A(W)(s)](0, .)$ is really close to one.

The recent work of [Nguyen et al., 2022] focused on reducing the entropy of the attention weights in order to identify, the relevant tokens for the classification and then provide a plausible explanation for the classification. These different experiments are not conclusive and do not allow to clearly identify the important tokens. Above all, these experiments show that playing on the entropy doesn't affect the performance in the final classification task. It seems that when reducing the entropy, the model compensates. [5] An interesting question is thus: How the model performs its compensation? How can we reduce the entropy of the attention weights without changing the performance in the classification task?

We provide here a possible explanation for this compensation. Based on Equation 1, we have :

$$\{[A(W)(s)](0, k)\}_{k\in\{0,...,T(s)-1\}} = softmax(\{\frac{< [Q_W(s)](0, .), [K_W(s)](k, .) >}{\sqrt{d}}\}_{k\in\{0,...,T(s)-1\}})$$

[5] this was one of the most important observations during my internship

where $< ., . >$ denotes the usual dot product. Then having a high entropy of $[A(W)(s)](0, .)$ is similar to having all the dot product $< [Q_W(s)](0, .), [K_W(s)](k, .) >$ similar for all k. Since $[Q_W(s)](0, .)$ is fixed, the result of these dot products will strongly depends on the geometry shaped by the rows of $K_W(s)$.

Having a high entropy, can happen when all the vectors $[K_W(s)](k, 0)$ are similar *i.e.* when the rows of K_W shape a narrow cone. If this is the case, then the effect of reducing the entropy will be to lower the quantity $SIM(K_W(s))$ and then break the cone shaped by the rows of $K_W(s)$. But if our thoughts are true and the classification depends on the cone shaped by the final embeddings (*i.e.* shaped by $W_{out}(s)$), in order to not change the final decision and don't lose any performance in classification task, the quantity $SIM(V_W(s))$ should increase in order to compensate for the break of the geometry of $K_W(s)$ (see 2).

0.5 Results

First, we will study similarities of our *Keys* and *Values* embeddings, because they seem to be the most important ones in the classification process.

This study was made for various models where we added more *attention block* (see Figure 3). We tested from $L = 1$ *attention block*, to $L = 5$ *attention blocks* (except for the E-SNLI dataset for which we have only tested $L = 1$ and $L = 2$). For every models with L attention blocks for every $0 < l \leq L$, we calculated $SIM(K_{W_{l-1}}(s))$ and $SIM(V_{W_{l-1}}(s))$ for a every sentence s in a given set of sentences. We then look at the mean over the different sentences.

The objective was to see the evolution of the metrics within the different models, and see the impact of adding attention layers, because in the different state of the art models, we are dealing with many attention layers.

Table 1 shows clearly that the matrices K_W and V_W tend to shape a narrow cone, and this cone gets narrower when we add more and more attention layers. Another observation is that the behavior of each attention layer doesn't seem to be affected by the addition of multiple layers, *i.e.* the columns of Table 1 have very similar values.

Our next move will be to check if this cone is a property of the class of the sentence. In order to do that for each class k of each dataset, we select a representative sentence s_k. This sentence will be a well classified sentence by the model. For each of these representative sentences, we will select at the end of the model (*i.e.* at the output of the last attention layer) : $w_{s_k} = W_{out}(s_k)(0, .)$, $v_{s_k} = V_W(s)(0, .)$ and $k_{s_k} = K_W(s)(0, .)$ our different $< cls >$ embeddings on the different spaces, for the different representative. Then for each of the remaining sentences s we will compute the following coefficients: $e_s^k(W) = cos(w_{s_k}, w_s)$, $e_s^k(V) = cos(v_{s_k}, v_s)$ and $e_s^k(K) = cos(k_{s_k}, k_s)$ If the different cones are a

	HatexPlain					YelpHat					E-SNLI	
	l=1	**l=2**	**l=3**	**l=4**	**l=5**	**l=1**	**l=2**	**l=3**	**l=4**	**l=5**	**l=1**	**l=2**
K_W												
L=1	0.713	-	-	-	-	0.578	-	-	-	-	0.657	-
L=2	0.691	0.683	-	-	-	0.6486	0.556	-	-	-	0.719	0.489
L=3	0.698	0.688	0.841	-	-	0.597	0.431	0.537	-	-	-	-
L=4	0.614	0.620	0.748	0.840	-	0.714	0.717	0.702	0.860	-	-	-
L=5	0.624	0.647	0.777	0.913	0.973	0.584	0.542	0.761	0.816	0.959	-	-
V_W												
L=1	0.542	-	-	-	-	0.372	-	-	-	-	0.524	-
L=2	0.510	0.740	-	-	-	0.409	0.511	-	-	-	0.182	0.746
L=3	0.605	0.688	0.88	-	-	0.461	0.371	0.494	-	-	-	-
L=4	0.5919	0.561	0.785	0.931	-	0.429	0.613	0.803	0.904	-	-	-
L=5	0.606	0.673	0.858	0.958	0.992	0.417	0.624	0.780	0.9023	0.972	-	-

Table 1: Mean of the similarity metrics on a large number of sentences on each dataset. Here the study is made on the KEY ($K_W(s)$) and VALUE ($V_W(s)$). The row represents the different models (different values of L). The columns represent the attention layer after which the metric is calculated (the different values of l).

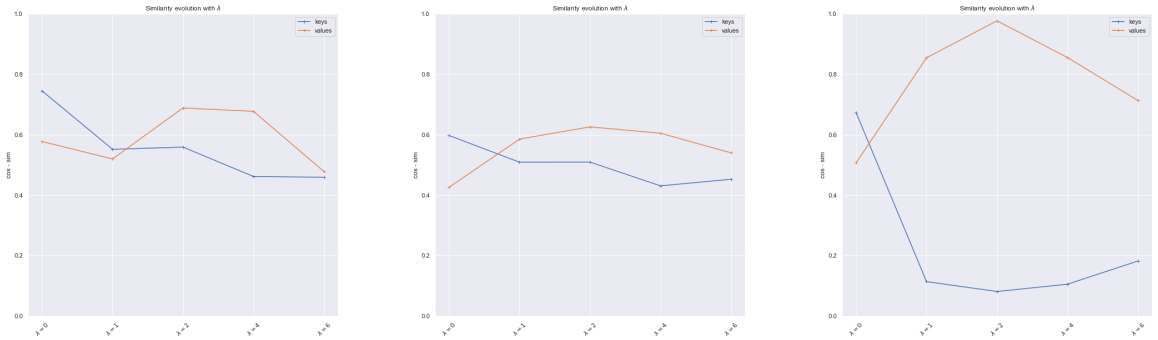
property of a sentence, then we should observe that for sentences s with the same class as s_k these different coefficients should be closer to one than for the sentences with different classes.

One may notice that we only proceed the comparisons between the $< cls >$ tokens of each sentence, but since we show that all the tokens of a sentence has similar embeddings, taking the embedding of the $< cls >$ token, rather than another token, shouldn't change in a significant way the results. Figures 6, 7 and 8, in the Appendix, show us the distribution of the different coefficients for the models with $L = 1$ and $L = 5$ ($L = 2$ for E-SNLI). On the three datasets we can clearly see that the coefficients $e_s^k(W)$ (for the embeddings) are separable, *i.e.* for each k the sentences s with class k tend to have higher values of the coefficient. For the other coefficients (for the keys and the values), the behavior appears when we add more layers.

These different results reinforce the idea that the attention mechanism tend to shape narrow cone, even in the projection of the embedding.

We will now try to show that these cones are decisive for the final classification of the model. In order to do that, we will proceed the entropy regularization of $A_W(0, .)$ as explained in [Nguyen et al., 2022], for the model with only 1 *attention block*. If the cone is decisive for the final classification, then the compensation explained in the section 0.4.3 should appear.

This regularization consists of adding a regularization term during the training part to the loss function



(a) HatexPlain (b) YelpHat (c) E-SNLI

Figure 5: Evolution of the similarity metric (red : V_W, blue : K_W) when proceeding the entropy regularization on the three different datasets. These figures represent the mean over many sentences. On this figure, the x axis, represents the regularization coefficient λ. We tested from left to right $\lambda = 0,\ 1,\ 2,\ 4,\ 6$

$\mathcal{L}(.)$ of the model. Then during the training of the model the loss function will be modified as follow for every sentence s :

$$\tilde{\mathcal{L}}(s) = \mathcal{L}(s) + \lambda \times H(A_W(0, .))$$

Where $H(.)$ denotes the Shannon entropy [Shannon, 1948], and $\lambda \in \mathbb{R}_+$ a regularization parameter which reflects the intensity with which we want to reduce the entropy. The larger λ, the more the entropy will be reduced.

Figure 5 show the evolution of $SIM(K_W)$ and $SIM(V_W)$ as measure as we increase the regularization parameter. We can clearly see that our thoughts were true, and we see that the regularization, by the entropy, tends to increase the cosine similarity of the values embeddings while it lowers the one of the keys, particularly for the E-SNLI dataset.

This last result allows us to understand how the model compensates. The model compensates for the parsimony of attention by increasing the anisotropy of the values embeddings. [6] Which tend to show that the geometry and particularly the cone shaped by the embedding is decisive for the classification.

The next question is: How the cones are shaped ? For a given sentence s, what are the different tokens, that help to construct this decisive cone.

Our hypothesis is, that some relevant token carry in them the information to classify the sentence, and

[6] In fact we showed that this property was true also for BERT [Devlin et al., 2019] architectures. This property was showed in the report of my 4 *th* year internship

	L=1	L=2	L=3	L=4	L=5
HateXplain	0.645	0.643	0.644	0.656	0.657
YelpHat	0.610	0.593	0.612	0.607	0.605
E-SNLI	0.567	0.690	-	-	-

Table 2: This tables contains the value of the AUROC between $\hat{\alpha}$ and α, for different models and different datasets

this is these tokens which will mostly participate to the construction of the final cone. In order to test this hypothesis, we introduce the following experiment.

For each token t of our vocabulary, we construct the following artificial sentence: $\tilde{s}_t :=< cls > \ t$. This sentence, will go through our model (with one or multiple attention blocks), we recover at the end of this model the output embedding of the $< cls >$ token *i.e.* the embedding which normally is used for the classification this vector will be denoted by u_t. Then for every word t of our vocabulary, we associate a new embedding u_t to it, which will be its representation in the "classification space". Then for every sentence s we associate the vector $cls(s)$ which is the classification token of the sentence. Then if $s(i)$ is the token at the position i in the sentence s, we construct $\alpha_{cos}(s) \in [-1,1]^{T(s)}$ as follow :

$$[\alpha_{cos}(s)](i) = cos(u_{s(i)}, cls(s)) \ \ \forall i \in \{0, T(s)-1\}$$

Then $[\alpha_{cos}(s)](i)$, will give us the correlation between the classification token of a sentence s and the representation of the token $s(i)$ in the classification space. If $[\alpha_{cos}(s)](i)$ is close to one, then we will assume that the token at position i strongly participated to the construction of the final cone. With our datasets, comes an annotation as we explained in the section the annotation is supposed to give us the relevant tokens of a sentence s for the classification task, then based on the work of [Nguyen et al., 2022] we will compare the values of the annotation $\alpha(s)$ and the values of our vector $\alpha_{cos}(s)$. First, for every sentence s we proceed:

$$\hat{\alpha}(s) = \frac{\alpha_{cos}(s) - min(\alpha_{cos}(s))}{max(\alpha_{cos}(s)) - min(\alpha_{cos}(s))} \in [0,1]^{T(s)}$$

This new vector $\hat{\alpha}(s)$ can be interpreted as the result of a binary classification which tries to predict the values of $\alpha(s)$ (*i.e.* which tries to predict the relevant tokens) we will then compare this two vectors by measuring the quantity $AUROC(concat_s(\hat{\alpha}(s)), concat_s(\alpha(s)))$. This $AUROC$ value was calculated for models with $L = 1$ to $L = 5$ (except for the E-SNLI dataset).

The table 2 shows us the evolution of the AUROC. We can clearly see that we rise the number of *attention block* the value of the AUROC tends to increase like the narrowness of the cone (previous results). This clearly show us how these models tend to work.

This last experiment allows us to show that as we add layers *i.e.* that we increase the anysotropic character of our model, the tokens that are annotated *i.e.* important in the classification task will see that their representation in the space generated by the attention model will be more correlated to the class cones of the sentence in which they are. This supports the idea that annotated tokens, i.e. those that are important for the classification, will participate the most in the construction of the class cone.

0.6 Conclusion

Through this study on the attention mechanism, we have shown how the attention mechanism tend to work. We have highlighted the construction of narrow cones by the model for text classification and that these cones were in fact a property of the class in classification tasks. These results come to consolidate the results of [Ethayarajh, 2019] and [Fosse et al., 2022]. More than that we tried to understand how these different cones are built. It appears at the end of this study that the cones seem to be carried by the tokens which are discriminating for the classification. Then to have access to these important tokens, we need to find a good way to represent each token in the final classification space. We provided here a very empirical way to construct such embeddings for our tokens, but the values of the $AUROC$ were a bit low. A possible extension for this study would be to provide for each token a good representation in the final classification space.

Sources

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization.

[Bibal et al., 2022] Bibal, A., Cardon, R., Alfter, D., Wilkens, R., Wang, X., François, T., and Watrin, P. (2022). Is Attention Explanation? An Introduction to the Debate. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3889–3900, Dublin, Ireland. Association for Computational Linguistics.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Bowman et al., 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

[Camburu et al., 2018] Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., and Blunsom, P. (2018). E-SNLI: Natural Language Inference with Natural Language Explanations. *Advances in Neural Information Processing Systems 31 (NeurIPS)*, page 11.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[Ethayarajh, 2019] Ethayarajh, K. (2019). How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. *Association for Computational Linguistics*, pages 55–65.

[Fosse et al., 2022] Fosse, L., Nguyen, D. H., Sébillot, P., and Gravier, G. (2022). Une étude statistique des plongements dans les modèles transformers pour le français. *TALN (traitement automatique des langues naturelles, 2022, Avignon France)*, page 10.

[Mathew et al., 2021] Mathew, B., Saha, P., Yimam, S. M., Biemann, C., Goyal, P., and Mukherjee, A. (2021). HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. *Conference on Artificial Intelligence (AAAI)*, page 9.

[Mikolov et al., 2013] Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representations. *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 6.

[Nguyen et al., 2022] Nguyen, D. H., Gravier, G., and Sébillot, P. (2022). Filtrage et régularisation pour améliorer la plausibilité des poids d'attention dans la tâche d'inférence en langue naturelle. *TALN (traitement automatique des langues naturelles, 2022, Avignon France)*, page 9.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[Sen et al., 2020] Sen, C., Hartvigsen, T., Yin, B., Kong, X., and Rundensteiner, E. (2020). Human Attention Maps for Text Classification: Do Humans and Neural Networks Focus on the Same Words? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4596–4608, Online. Association for Computational Linguistics.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

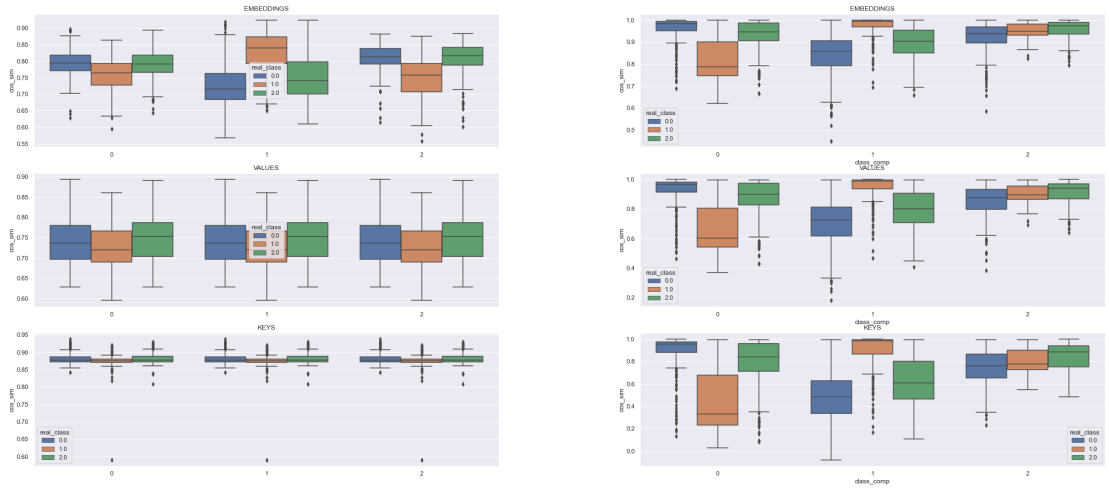
[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, page 11.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

[Xiong et al., 2019] Xiong, Z., Shen, Q., Xiong, Y., YijieWang, and Li, W. (2019). New Generation Model of Word Vector Representation Based on CBOW or Skip-Gram. *Computers, Materials & Continua*, 60(1):259–273.

Annexes

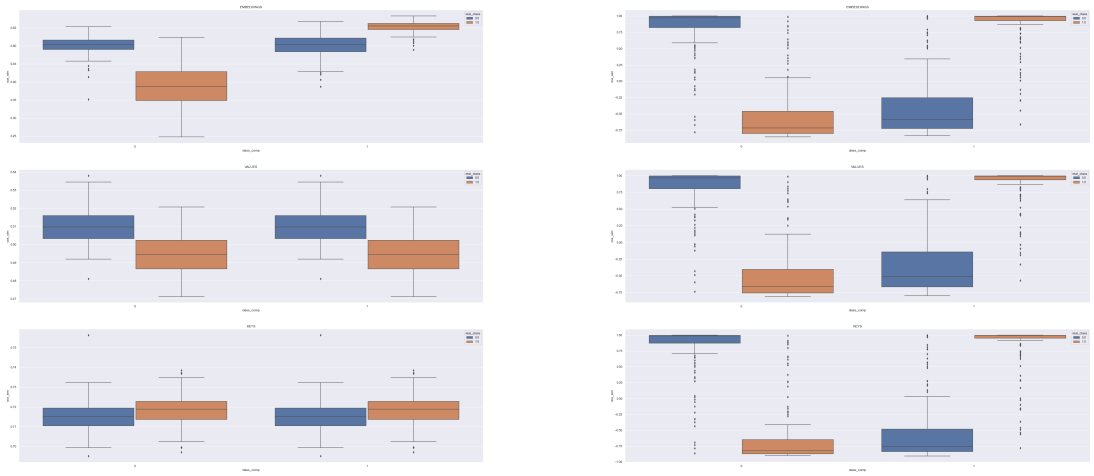
Figures



(a) Hatexplain : $L = 1$

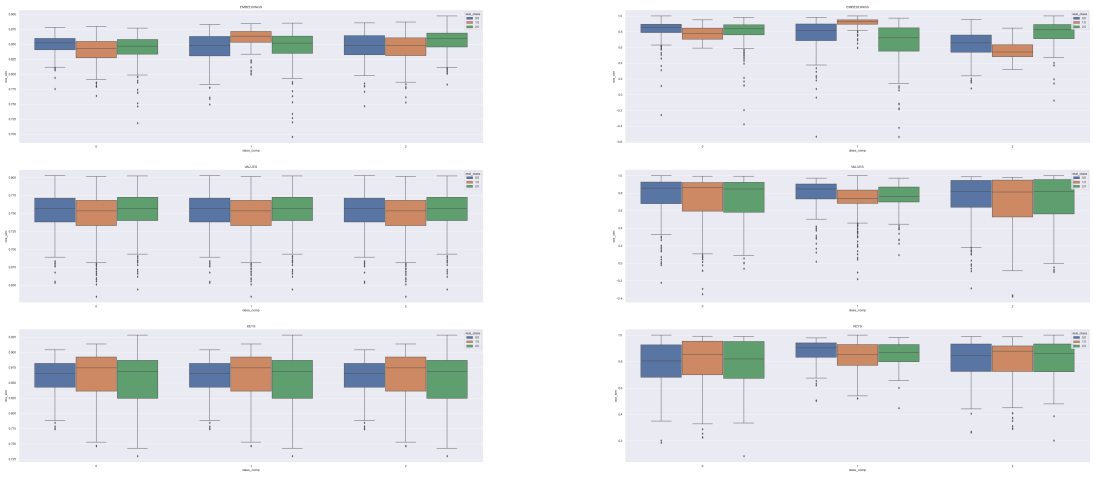
(b) Hatexplain : $L = 5$

Figure 6: Distribution of the coefficients $e_s^k(W)$ (top), $e_s^k(V)$ (middle) and $e_s^k(K)$ (bottom) for 900 sentences on the hatexplain dataset. The x-axis represents the value of k, the class with the one we compare. The color of each box represents, the real class of the sentence.



(a) yelphat : $L = 1$ (b) yelphat : $L = 5$

Figure 7: Distribution of the coefficients $e_s^k(W)$ (top), $e_s^k(V)$ (middle) and $e_s^k(K)$ (bottom) for 300 sentences on the Yelphat50 dataset. The x-axis represents the value of k, the class with the one we compare. The color of each box represents, the real class of the sentence.



(a) esnli : $L = 1$ (b) esnli : $L = 2$

Figure 8: Distribution of the coefficients $e_s^k(W)$ (top), $e_s^k(V)$ (middle) and $e_s^k(K)$ (bottom) for 900 sentences on the E-SNLI dataset. The x-axis represents the value of k, the class with the one we compare. The color of each box represents, the real class of the sentence.