# Final Master Project: Explainability of Language Models applied to Question Answering


**Final Master Project**

**Marcos Martínez Galindo**

Research project for the

Master in Language Technologies

National Distance Education University

Directed by

**Prof. Dr. D. Álvaro Rodrigo Yuste**

June 2021

# Abstract

Breve resumen del trabajo realizado y de los objetivos conseguidos en inglés.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1. Motivation

With the increase of the data available and the computational power, Artificial Intelligence systems are more often used in the world, finding them in almost any sector, from banking to health. The Natural Language Processing is one of the main areas and one of the most used nowadays, helping companies to automatically process documents on any type (such as bank statements, salary slips, medical examinations and so on) to take decisions that may be critical (like credit grant).

But these systems learn from data in an statistical way, and sometimes fail. Ever year the performance of these systems is improved, but these use to be reflected also in the complexity of understanding how these systems work, being almost impossible for a human to understand what is going on inside a model and thus, to understand why a model gives an specific output.

This lack of understanding in the model performance is critical, preventing companies to use them to take critical decisions that can not be explained. Imagine a person asking for a credit grant that is denied, without any explanation why.

As said before these systems learn from data that have been recollected for years. And this data may be biased against, for instance, people of a specific gender, race, etc. If humans can not understand the model decisions can not know if this have been due to this bias or not, preventing them to avoid discrimination.

To solve this, new regulations control the usage of these systems, for instance in the *GDPR* the "Right to Explain" is a must have for any system, what could lead to forbid the usage of these systems.

Because of all of this, the ability to explain what is going on inside the models and to explain why a model has given an output is a really important task. This work studies

the most important tools for explainability applied to language models, studying the state of the art of this technologies and checking if these tools can be applied to specific problem such as question answering and if there are indeed useful to understand the behavior of the systems.

## 1.2. Proposal and goals

In this work some of the most famous tools for explainability are going to be tested over a language model trained for a multiple choice problem, more specific a *BERT* model trained over the *RACE* dataset is going to be used.

The state of the art of the language models, the question answering task and the explainability is going to be reviewed, and finally some tools are going to be tested with a model trained for a multiple choice problem.

## 1.3. Document structure

**Chapter 1. Introduction.** In this chapter the main motives for the realization of this essay, the proposal and the goals to achieve are presented, as well as the document structure.

**Chapter 2. State of the art.** The discipline and the task are reviewed in this chapter, including a brief introduction of the explainability, the language models and the question answering task and the state of the art of this areas. At the end of this chapter, the state of the art of explainability applied to Natural Language Processing tasks is reviewed.

**Chapter 3. Methodology.** The model to be used is trained and the different methods and algorithms used to solve the problem are applied to the multiple choice problem.

**Chapter 4. Results.** After knowing the different methods and algorithms being used, the experiments with their results are presented in this chapter.

**Chapter 5. Conclusions and future work.** The conclusions of the review of the state of the art in addition to the conclusion of the usage of the different methods and to the future work that could be done after this project.

# Chapter 2

# State of the Art

Although both language models and explainability are young techniques, both of them have awaken the interest of the researchers and, therefore, there are a lot of works regarding to these areas of research.

In this section, the state of the art of the language models, the question answering task and the explainability is being reviewed, from the history to the most novel methods, seeing the evaluation methods and *datasets* available and the most interesting papers.

## 2.1. Language models

In 2001, Joshua Bengio et al. proposed the first neural language model, based on a feed-forward neural network.(Bengio et al., 2001) However, it wasn't until 2013 that neural networks begin to gain relevance in the world of *NLP*, and it was due to the *word embeddings* and the new neural architectures, like *Recurrent Neural Networks* (*RNN*).

The *word embeddings* are vectorized representantions of the words contained in a document, and since its begining have gained a lot of importance in the world of the *NLP*, because they are able to capture the context of the word within the text, improving the way the model understands the text. This way, a polysemic word has not the same representation in different contexts. One example of this techinque is *Word2Vec*, probably the most used technique by the researchers to learn the *word embeddings* of the text, and was developed by Tomas Mikolov in 2013.(Mikolov et al., 2013).

When dealing with temporal series or data sequences, neural networks are not able to model the temporal dependency, as they don't have knowledge about the previous sequences. To solve this, a new variety of neural networks was developed, the *RNN*, that have the ability to process data sequences by saving memory of the already processed ones. In *NLP* this is really important, because in natural language texts a single word is not enough to capture its meaning, also the previous words (or sentences) are needed, so the ability to save "memory" about the previous words improved the performance of

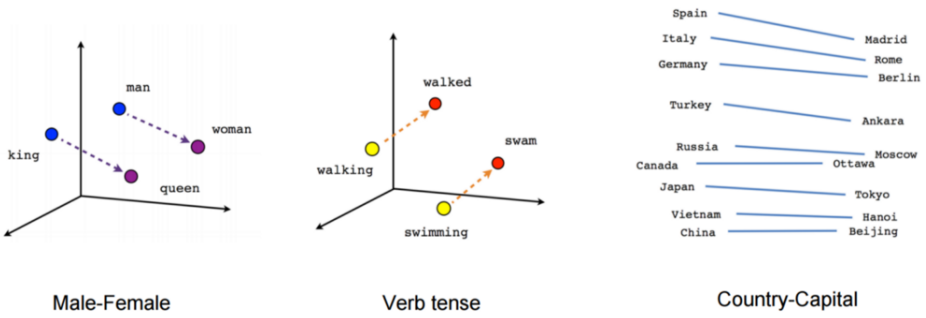



Figure 2.1: Word Embeddings

neural networks in almost every *NLP* task. Thus, a lot of versions of neural networks have been developed, like the *Long Short-Term Memories* (*LSTM*), that are very used in *NLP* because they save the memory during more time than other versions, increasing the window size of the sequences being seen.

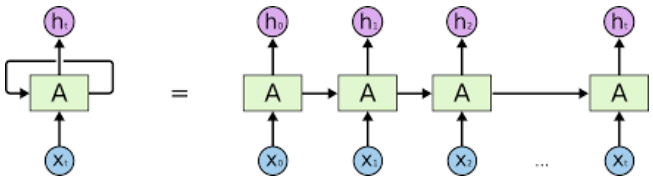



Figure 2.2: *LSTM* Recurrent Neural Networks.

Other architecture that has revolutionized the research world in *NLP*, contributing to the appearance of new language models is the *encoder-decoder* architecture, based on two different parts, the *encoder* that codifies the input, getting a vectorized representation that is transmitted to the second part, the *decoder*, that from this codified sate generates the output. Usually, a sentence is given as input and the output is a new sentence, so that this architecture is also known as *sequence to sequence*. In its beginning it was used to *machine translation* task, having as input a text in some language and the translation as output.

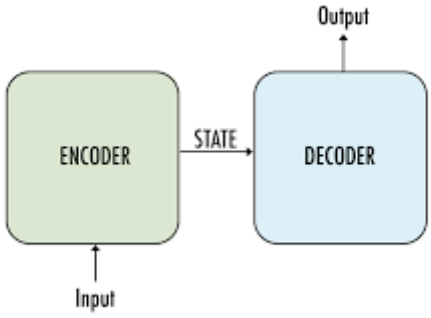



Figure 2.3: *Encoder-Decoder* architecture.

One of the main problems of this architecture is the need of codify the whole input before the output generation, that is done by looking at the whole input. The *attention* mechanism was developed to solve this, giving as input for the *decoder* not only the codified state of the input (as in the classic *encoder-decoder* architecture) but also the hidden states of the *encoder*, that now process the input splitted, making possible to look only at one part of the input and not the whole input as before.(Bahdanau, Cho, y Bengio, 2014)

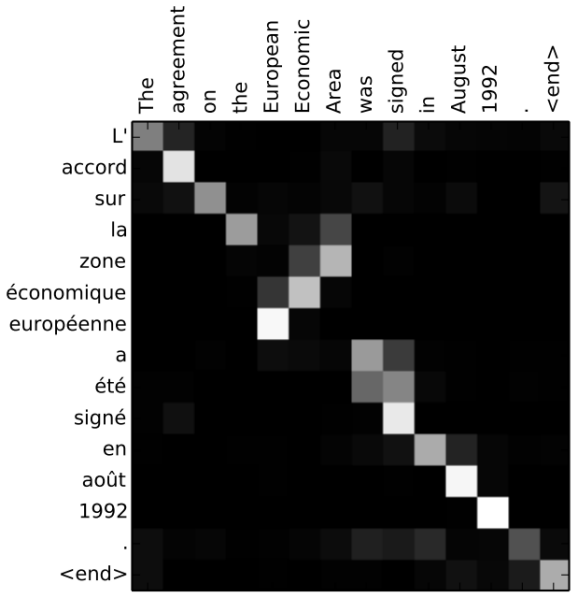



Figure 2.4: Example of the *Attention* mechanism in machine translation. It can be seen as the *Attention* mechanism focus only on the words being translated and the relevant ones instead in the whole sentence.

Three years later, in 2017, probably the most important article in the history of *NLP* was released: "Attention is all you need".(Vaswani et al., 2017) In the article, Vaswani et al. propose a novel neural architecture based only on layers using the *Attention* mechanism, the *Transformer*. Besides, the *Transformers* use a variation of the *Attention* mechanism called *Self-Attention*, that allows to take as input other positions of the self input, which makes possible to use other words of the same input to codify each word, as can be seen in Figure 2.5.

In the *Transformer*, the output of different *Self-Attention* layers are concatenated, creating a multi-dimensional *Attention* representation, what is the *Multi-Head Attention*. This way, each head can learn to codify the same word in different ways, what allows the model to capture more information of the word. Although it can be considered as *Transformer* every architecture designed to process a set of words with the *Attention* mechanism as the unique way of interaction between them, the basic structure of the *Transformer* is based on:

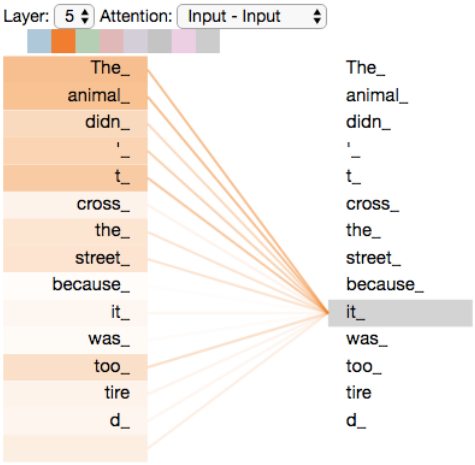



Figure 2.5: *Self-Attention* mechanism.

- *Self-Attention* or *Multi-head Attention* layer.
- Normalization layer.
- *Feed-forward* layer.
- Normalization layer.
- Residual connections.

With this, an *encoder-decoder* architecture is developed, as shown in Figure 2.6.

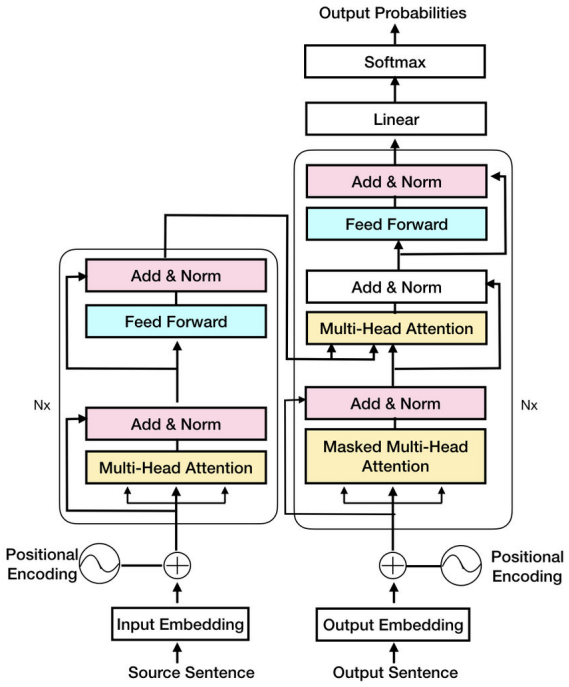



Figure 2.6: *Transformer* architecture.

Another one of the key points to the massive expansion of the language models has been the training techniques used. Along the history of the language models, they used to be unidirectionals, this is, to predict the next word they used only the words before. But this changed with the arrival of *BERT*, *Pretraining of Deep Bidirectional Transformers for Language Understanding*. (Devlin et al., 2018) Instead of predict the next word, *BERT* masks randomly some tokens in the input, and trains by trying to predict the original words that have been masked, using both left and right sides of the sentences (bidirectional), which allows the model to capture more information while codifying the tokens. This is known as *Masked Language Models*, *MLM*.

Although the most common way to work is dealing with words, the language models can work in different ways, like for example using *n-grams* and using smooth techniques to deal with *n-grams* neer seen before(Kneser y Ney, 1995). Another ones, like *ProphetNet* (Yan et al., 2020), try to predict not only the next token, but the $n$ following tokens.

So, the basic idea behind the language models is to predict the next token, but they can be used also for other *NLP* tasks such as *Question Answering*. To achieve this, different additional layers are set on top of the language model, trained to solve the specific task, taking as input the output of the language model, this is, the original input tokenized.

Therefore, there are two different parts, the language model and the specific layers, and there are also two different types of training, the one used to train the language model (without the specific layers), that can be trained using massive amounts of texts, like the *Wikipedia*, and the one used to train the model to the specific task, which requires a specific *dataset*, which is known as *fine-tuning*.

### 2.1.1. Main Language Models for Question Answering

Since the release of *BERT* the number of papers and models published has suffered an exponential grown, Figure 2.7, most of them using the same architecture than *BERT*, or with little variations. So, after a research of the state of the art of language models, it has been developed a brief review of some of the most interesting models.

#### BERT

As said before, the "boom" of the *LM* started with *BERT*, *Pretraining of Deep Bidirectional Transformers for Language Understanding*(Devlin et al., 2018). In this article, Devlin et al. propose a novel architecture based on the classic *encoder-decoder* but, in this case, made only with *Transformers*. There were released two different versions of *BERT* depending on its size: one with 12 layers and 110 million parameters (the same as

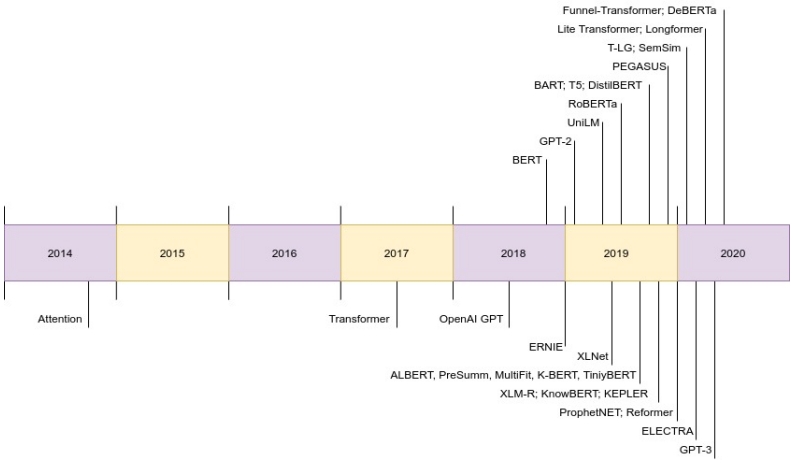



Figure 2.7: Chronological view of a sample of language models. It can be seen that since *BERT* the number of language models released has grown exponentially.

the *OpenAI GPT* for comparison purposes); and one with 24 layers and a total number of 340 million parameters.

The *WordPiece* embeddings ((Wu et al., 2016)) are used, with a total umber of 30.000 *tokens* in the vocabulary. The first *token* of each sequence (understanding as sequence not a sentence of the text, but the input given to *BERT* than can be up to two sentences) is a *CLS token*, sentences in the input sequence are separated by a *[SEP] token*.

In addition to this, a (not) new training is used, in which some random *tokens* are masked (replaced by a *[MASK] token*), and the model has to predict which one was the original *token*. Altough this method started to be famous thanks to *BERT* it wasn't invented by Devlin et al., but it was known as *Cloze* task in the literature (Taylor, 1953). In the article, the authors explain that while training *BERT* a 15 % of the *tokens* is 'masked". When masking a *token*, there are 3 possibilities:

1. The *token* is replaced by a *[MASK] token* 80 % of the time.
2. The *token* is replaced by a random *token* 10 % of the time.
3. The *token* is not changed 10 % of the time.

But this is not the only way *BERT* is trained. It is also trained for another task, the prediction of the next sentence, *NSP* (Next Sentence Prediction). In this task, two sentences are selected from the text ($A$ and $B$). 50 % of the time $B$ is the sentence that really follows $A$, meanwhile a 50 % of the time is not. In this task $BERT$ has to classify if $B$ is the next sentence following $A$ or not.

### T5

*T5*, created by Raffel et al. in Google, in 2019.(Raffel et al., 2019) is a model proposed in an extense research article with multiple tests of different architectures based on

*Transformers* and with a mask training, the same used by *BERT*.

Besides the high-quality research presented, *T5* does not propose any novel architecture, but due to its high size and its training over a really big *corpus* obtained from the *World Wide Web* (known as *C4*, *Colossal Clean Crawled Corpus*), *T5* obtains state-of-the-art performance in multiple *NLP* tasks, such as *Question Answering*.

### GPT-2/GPT-3

*GPT* architectures, (Openai et al., )(Radford et al., 2019)(Brown et al., 2020), are a group of language models based on an architecture based only on *decoders*, without the encoder that is habitual in language models. These decoders layers are trained predicting next words, without a masked training, as it used to be in the past, which is known as "from left to right". This way, the model can only see the words at the left of the token being processed, loosing some of the context, but the quality of the generated text in *Natural Language Generation* (*NLG*) tasks is usually of a higher quality, because in runtime while generating new text, the model can only see the text at the left of the word.

*GPT-2* architecture caused quite a stir among all the researchers in *NLP* (and also in the news), due to the quality of its generated text. But it also demonstrated that *LM* are able to generalize and solve tasks without a specific *fine-tuning* for the task. This way, *GPT-2* was trained on a huge *corpus*, but it was not trained for a specific training such as *QA*. Even when the results are not as good in the benchmarks as the ones obtained from other models trained specifically for the task, results are really good, especially in *NLG* tasks, which is surprisingly due to the fact that the model has not been trained for that.

The *dataset* used to train *GPT-2* is known as *WebText*, and was generated from posts of the social network *Redit*. To assure a good quality of these posts, only those who had a high punctuation were used, obtaining this way a set of texts well-written and covering different topics.

But, again, it wasn't trained for a specific tasks. Nevertheless, posts in *Redit* are written by real people interacting with each other, so the model has been able to learn how to solve different tasks of *NLP*. For example, in these posts some users made a question that other users answered, and this way the model learned to answer questions.

Other interesting example is for the *Text Summarization* (*TS*) task, which consists on generate an abstract from a text. In these posts, users used to add a brief summary at the end of the post, tagged with the acronym *TL;DR:* (Too Large; Didn't Read:).

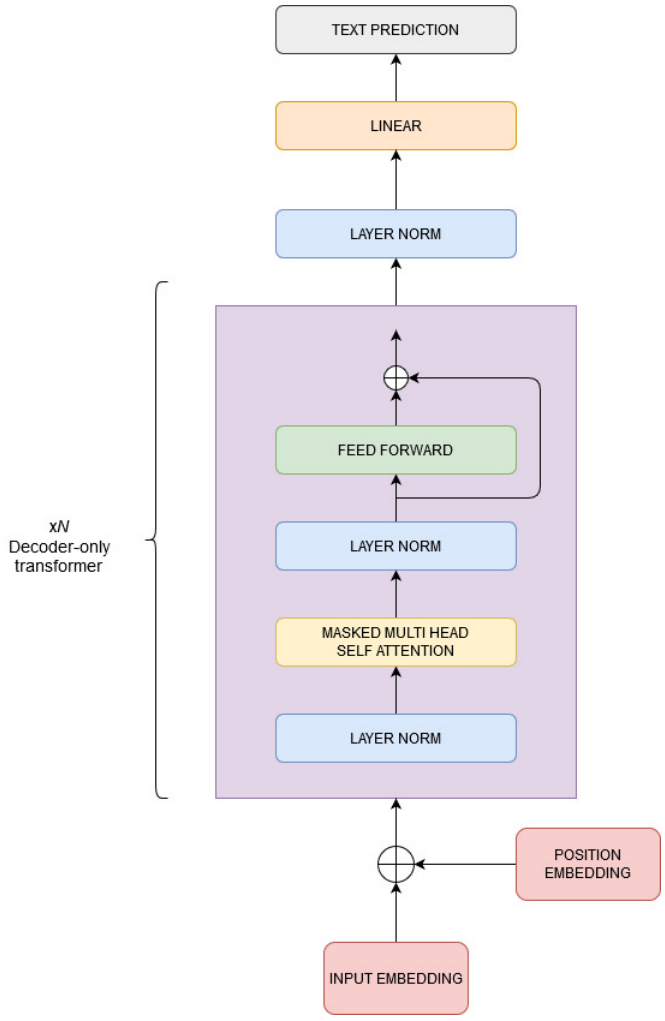



Figure 2.8: *GPT-2* architecture.

Researchers found that if this token is added to the model at the end of the input, the text generated by *GPT-2* is the summary of the input, solving the *TS* task.

When *GPT-2* was released its large size was surprised (1.542 millions - 48 *decoder* layers), which makes possible to understand the text and to generate a high quality output. But now the size of *GPT-2* is not even close to the largest models used in *NLP*.

Its "big brother", *GPT-3*, its the largest model by the date of this article, with 175.000 millions of parameters, what allows it to understand the language in a way never seen before. Its able to solve any task in any domain without needing to adapt it with a specific *fine-tunnning*, which is known as *zero-shot*.

Its also able to solve, not only *NLP* tasks, but also even arithmetic operations up to 3 digits, something really impressive that demonstrates its knowledge beyond the languages.

### Turing NLG

*Turing NLG* or *T-NLG*(Rosset, 2020) its a model developed by *Microsoft* and, till the release of *GPT-3* was the largest model published, with 17.000 millions of parameters (10 times less than *GPT-3*). Nevertheless, its still a really massive amount of parameters, which allows *T-NLG* to reach state-of-the-art performance in almost any *NLP* task, such *QA* or *TS*.

Due to it large size, the *Microsoft* team developed a *Deep Learning* library (*DeepSpeed*) and a new optimizer (*ZeRo*) that allow the parallelization in the optimization and reduce the resources needed by the model to make possible the training of large models in a more efficient way, with less resources.

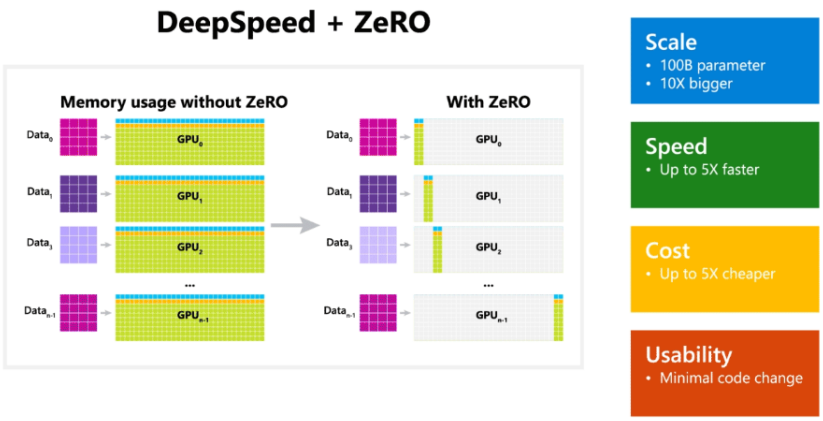



Figure 2.9: *T-NLG*, *DeepSpeed* and *ZeRo*.

In addition, *T-NLG* has demonstrated that, as large is the *LM* the less of data is needed to learn, improving results of other architectures using less amounts of data. Despite of this, *T-NLG* trained with practically any *dataset* available for different tasks, which gives the model a good knowledge of different domains. This also makes possible for *T-NLG* to use different types of documents, such as *e-mails* or *PowerPoints*.

Unfortunately, code has not been released, but the results published by *Microsoft* are encouraging.

### BART

*BART* is a model based on the standart architecture *sequence-to-sequence* with *encoder-decoder* layers based on *Transformers*, Figure 2.10(Lewis et al., 2019). Its compound of a *bi-directional encoder* (like *BERT*) followed by a *left-to-right decoder* (like *GPT*), which allows it to learn the context in a good way (thanks to the *bi-directional encoder*) and to generate high-quality text (thanks to the *left-to-right decoder*).

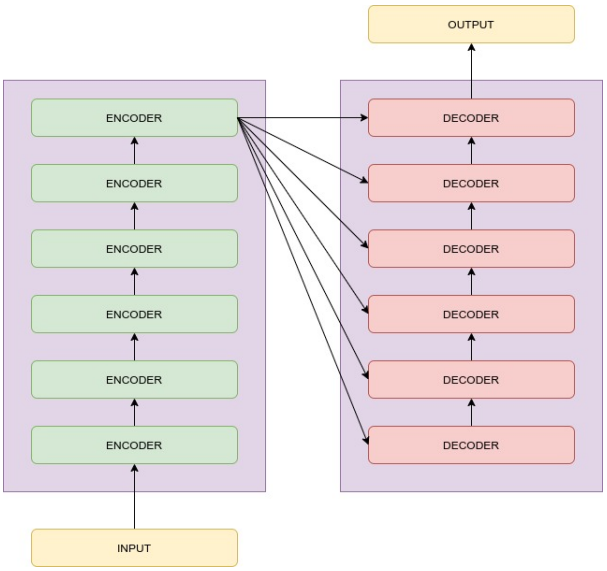



Figure 2.10: *BART* architecture.

The training used for *BART* is based on the mask training used by *BERT* but with a few variations. The model is optimized based on a loss function obtained between the output of the *decoder* and the original text, that is "corrupted" with different transformations to make a noised input, forcing the model to capture the language information in an extremely way to understand the noisy input.

Among the different ways of "corrupting" the text, some of the are:

- *Token* masking.
  Same as *BERT*, it masks random *tokens*.
- *Tokens* removal.
  In addition to the mask of different *tokens*, *BART* also remove them, thus it has to learn not only to replace the masked *tokens* with the original ones, but also to recognize is some *tokens* are missing and, in that case, to predict which one was the original *token*.
- *Spans* masking.
  *BART* not only mask single *tokens*, but also text *spans* are completely masked with a single *[MASK] token*. This way, *BART* has to learn if the mask *token* was a single word or a text *span*.
- Sentence permutation.
  Sentences are randomly shuffled to make the model to learn the real order of the sentences.
- Document rotation.
  A random *token* is chosen to rotate the document in a way that it begins with that token, making *BART* to learn to recognize the real start of the document.

This way, *BART* trains with highly noised documents and it has to learn to reconstruct them, allowing it to reach a high understanding of the language and to execute different *NLP* tasks, like *QA* or *TS*.

### ProphetNet

Another interesting approach is the one proposed by Yan et al.(Yan et al., 2020) *ProphetNet* is a novel architecture based also on *encoder-decoder* but, instead of predicting just the next *token* its trained to predict the $n$ next *tokens*.

This way, authors say that the overfitting in near words is avoiding as also its the underfitting in far away ones, because in traditional *LM* to predict the next *token* the model focus only on near words, loosing therefore the information provided by *tokens* far away.

But in some *NLP* tasks such *TS* and when dealing with large texts all the context is important for the result, making *ProphetNet* an interesting approach.

### ELECTRA

Clark et al. propose *ELECTRA* (Efficiency Learning an Encoder that Classifies Tokens Replacements Accurately).(Clark et al., 2020) Instead of using the mask training like *BERT* (and so others *LM*), *ELECTRA* uses a novel training way.

Instead of replace random *tokens* with a *[MASK] token* (as *MLM* use to do), *ELECTRA* replace them with plausible alternatives by using a "generator" (another *MLM*). This way the "discriminator" tries to recognize the replaced *tokens*, classifying them as "original" or "replaced", Figure 2.11. This training reminds to the Generative Adversarial Networks used in Computer Vision tasks, in which two neural networks compete, one of them generates fake images and the other one has to recognize the fake images from a set with real samples.

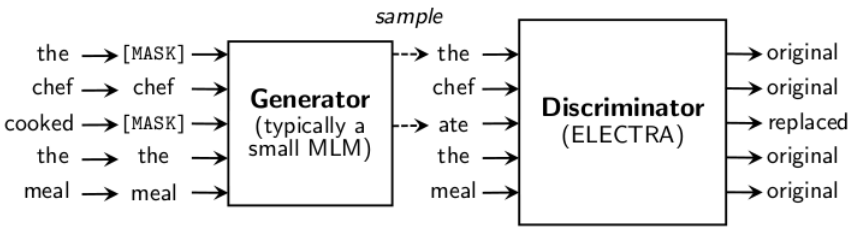



Figure 2.11: *ELECTRA* architecture.

Therefore, *ELECTRA* trains with every single *token* within the document, not only with those that has been masked like the rest of *MLM* do. This makes that *ELECTRA* is able to understand the language before than other *LM* in a more efficient way Figure 2.12.

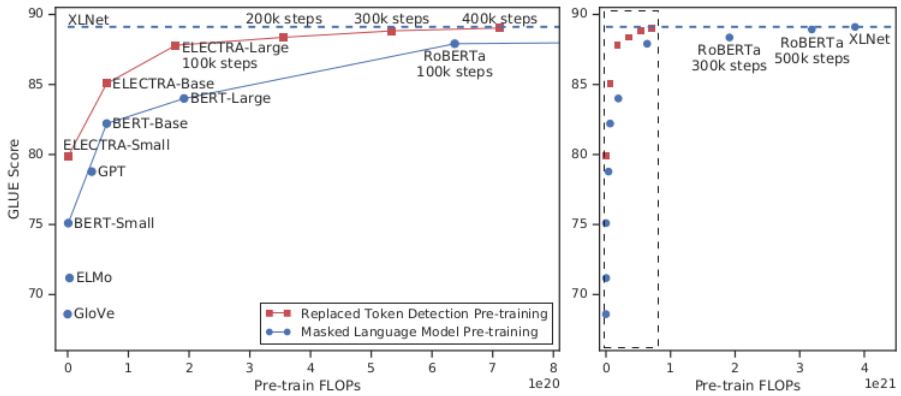



Figure 2.12: Comparison of *ELECTRA* with other *LM*.

This training has allowed that *ELECTRA* reaches state-of-the-art results and, as can be seen in Figure 2.12, in an efficient way.

## 2.2. Question Answering

Since the beginning of the *NLP*, one of the task most studied and researched is the ability to retrieve information based on a simple query, which is know as Information Retrieval (*IR*). This helps people to find any kind of information in the vast amount of data that is being recollected day to day. For example, search engines like *Google*, are not more than a system of *IR* able to find the documents (web pages) that are more related to the query.

But most of the time what people wants is not the whole document but a concise answer to a specific question. This is what the Question Answering tries to solve. Given a question over a context, the *QA* systems try to gives an answer to the user. This context use to be a document that could have been retrieved by an *IR* system, but it's only a document, not a collection of them.

This has made possible to search engines and intelligent assistants to improve its performance and to understand the user, improving therefore the user experience with this kind of systems. In Figure 2.13 it can be seen how the search engine *Google* answers a given question with a concise answer.

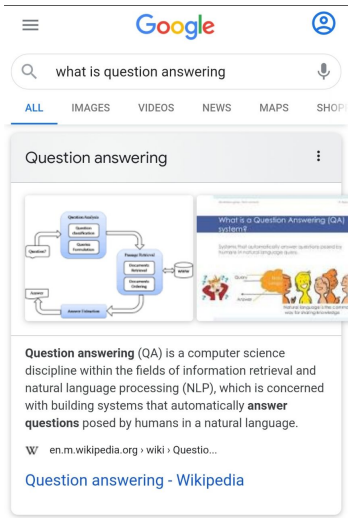



Figure 2.13: Question Answering system working in the search engine *Google*.

These systems are able to give the answer not only in a written form, but also verbally, and giving also the reference of the document where the answer has been found, in this case *Wikipedia*. As these systems result to be really useful and interesting, both the research and the company fields focused on this task. For instance, the *QA* task was included in the *TREC*, Text REtrieval Conference, or the released of the *Watson* system by *IBM*.

Along the history of the *QA* systems there have been proposed different architectures for tese systems, but all of them share some modules that are common in all the systems, as Gonzalez pointed (González, 2003). These basic modules can be seen in Figure 2.14.

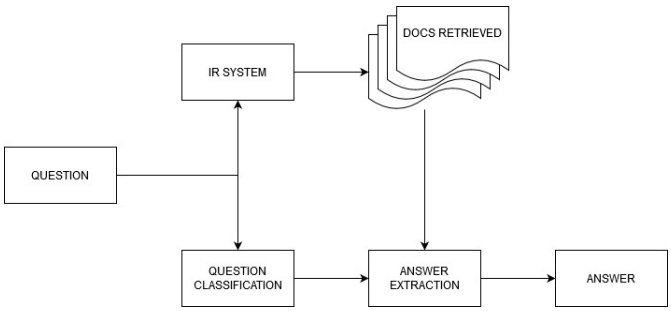



Figure 2.14: Question Answering common modules.

The basic workflow in *QA* systems is as follows:

1. An *IR* system retrieves the most relevant documents, meanwhile the question is analyzed and classified. The question is classified into different types (e.g.: yes/no question, year-expected, etc) to know what kind of answer is expected (a name, a year, an affirmation or negation, etc).

2. Once it is known the type of answer expected by the question and the most relevant documents have been retrieved, the question is searched within each of the documents. If the answer is found it can be returned to the user or it can be weighted in case there are more than one answer found.

These modules are expanded (or reduced) in some approaches, like for instance in (Ahmed y Anto, 2016), where a sub-module is added to the *IR* module. This new sub-module, the *passage retrieval*, is in charge of retrieve the passage within the document that is related to the question. This way, the *answer extractor* module has to look for the answer not in the whole document, but in the related passages. This can be good, less text means less computational and temporal cost, or bad(less text could mean worse performance, because if the *passage retrieval* fails and do not retrieve the passage the answer is located in, the *answer extraction* is not going to be able to find it. But, if the module works as expected, less text will mean also less noise to the *answer extraction* module, which could improve the performance of the system.

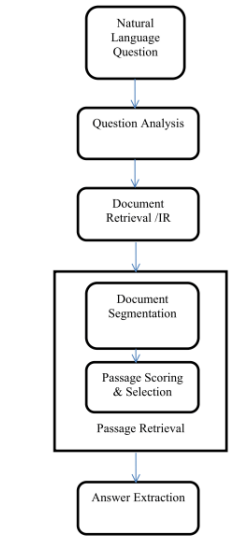



Figure 2.15: Workflow proposed by Ahmed.

This *passage retrieval* module was used in multiple approaches, like the one proposed in (Pablo-Sánchez, 2020), and it was a single task in *TREC* before the *QA* task.

But, with the appearance of the *Transformer* architecture, these modules has changed and modules like *question classification* has even been removed. This module classified the question to know what kind of answer was expected. Along the different *TREC* conferences there have been proposed a los of taxonomies of question and answers types, like the proposed by Moldovan et al. in (Moldovan et al., 2001) for the *TREC-8*, the one

| PERSON | PLACE | DATE |
|---|---|---|
| NUMBER | DEFINITION | ORGANIZATION |
| DESCRIPTION | ABBREVIATION | KNOWNFOR |
| RATE | LENGTH | MONEY |
| REASON | DURATION | PURPOSE |
| NOMINAL | OTHER | |

Table 2.1: Taxonomy proposed by Radev et al.

proposed by Li and Roth for the *TREC-10*(Li y Roth, 2002) or the proposed by Radev et al. in (Radev et al., 2002), Table 2.1.

There are also different types of *Question Answering* systems based on the structure of the problem. Some of the time the system is given just a document and a question, and it has to look for the answer to the question within the document, extracting the *span* of text that answers the question and this is why these systems are known as *Extractive Question Answering*.

However, sometimes there are a few options and the model has to choose which one is the correct answer to the question, what is known as *Multiple-Choice*.

But both of them have the same objective, to find an answer to a question. The first type, the *Extractive QA* has to find for the *span* of text that answers the question. This way, it is able to find the question within the text but, on the other hand, it can only answer with the *span* extracted and it is not able to answer applying *common-sense* and, for instance, they're not able to answer with a "yes" or "no", they can only extract the span more related to the expected answer by the question.

In the second type, the *Multiple-Choice* problems are able to apply some *common-sense*, being able to answer "yes/no" questions, relating countries with cities, languages and so on, but they can only choose between the options given, without being able to find the answer by themselves.

### 2.2.1. Evaluation of QA systems

The same way that there are different types of *QA* systems, there are also different evaluation metrics. The beginning of the *QA* systems can be traced to 1999, when the *TREC-8* has a specific track for *QA*. In this track, competitors had to propose different approaches to give a concise answer to a given question that was chosen from different sources, from question asked by the competitors their-selves to question obtained from the *FAQFinder* system.

At the end, the systems were evaluated, in this case a manual evaluation was carried out, this is, were humans the ones that evaluated. As is explained in (Voorhees, 2000), several problems were found while evaluating the systems, with even humans doubting about which one was the correct answer.

For instance, one of the doubts was if it would be considered as right in questions about a person just the full name, or if the name or the last-name should be considered as right, or if a list of different names is returned, in which the correct name is contained, but the user don't know for sure which one is.

Because of this, they came to the conclusion that, to an answer being considered as right, this should answer the question without any doubt, this is, without having to read the document to know the full answer.

Some of the criteria used to consider the answer as right are the next ones:(Allam y Haggag, 2012)

- Relevance: The answer returned should be a response to the question.
- Correctness: The answer should be factually correct.
- Conciseness: The answer should not contain extraneous or irrelevant information.
- Completeness: The answer should be complete (not a part of the answer).
- Justification: The answer should be supplied with sufficient context to allow a user to determine why this was chosen as an answer to the question.

Based on these concepts the answers are classified into different types, as explained in (Allam y Haggag, 2012) and (Pablo-Sánchez, 2020):

- Correct answer.
- Failed answer.
- Inexact answer (irrelevant content or missing data).
- Unsupported answer (not supported by other documents).

With these criteria the doubts commented before are solved, for instance, the fact that the answer is not complete (the problem of the names), the fact that the answer alone must be enough for the user without other sources or that the answer should be concise (for example, not returning a list of multiple names).

These evaluation metrics have also evolved and nowadays, even though the manual evaluation carried out by humans is still used in some articles, the most common is to evaluate the systems in an automatic way, like the *precision*, the *recall* or the *f1-score*. But other metrics less known have been used, like the ones used in the conferences *TREC-8* and *TREC-11*, the *Mean Reciprocal Rank (MRR)* and the *Confidence Weighted Score (CWS)* respectively, that are defined as:

$$MRR = \sum_{i=1}^{n} \frac{1}{r_i} \tag{2.1}$$

Where $n$ is the number of questions and $r_i$ is the *rank-value* of the first correct answer for the $i$th question.

$$CWS = \sum_{i=1}^{n} \frac{p_i}{n} \tag{2.2}$$

With $n$ being the number of questions and $p_i$ the *precision* of the answers from the 1 to the $i$ in a sorted list of answers.

Meanwhile the *Multiple-Choice* problems are easily assessable, because can be treated as classification problems, but the *Extractive QA* systems are more complicated to evaluate, because there can be more than one *span* within the text could be a correct answer from the question. Systems also can return the correct labeled *span* but with some added words or without all of them, but being still a correct answer.

Anyway, to be able to evaluate these systems in an automatic way, several *datasets* were made.

### 2.2.2. Datasets of QA

To be able to train these systems, there is need a group of questions with their correct answer and the contest, as well as to evaluate them. Therefore, a lot of *datasets* have been developed. Some of the most famous are the next ones:

#### SQuAD

The *SQuAD* (Sanford QUestion Answering Dataset)(Rajpurkar et al., 2016) is a *dataset* created by Rajpurkar et al. in 2016 for *Extractive QA*. It consists of more than 107.785 questions asked over 536 different *Wikipedia* articles by crowdworkers. Given a *Wikipedia* article they have to post a question about the text that the answer is a *span* of text included in the article.

To collect the *dataset* the authors used Project Nayuki's *Wikipedia's* internal *PageRanks* to obtain the top 10000 articles of English *Wikipedia*. Then, they select 536

random articles from them and removed figures, tables, short paragraphs, etc. The finally get a set of 23.215 paragraphs for that 536 articles of different topics. Finally they splitted the *dataset* into a training set (80 %), a development set (10 %) and a test set (10 %).

Once the articles had been recollected, they employed crowdworkers (like *Amazon Mechanical Turk*) to create the questions and its answers.

Authors then analyzed the question types, the difficulty of the question and the degree of syntactic divergence between the question and the answer. Table 2.2 shows the different question types and its percentage of appearance in the *dataset*. A complete and official analysis of the *dataset* can be found in (Rajpurkar et al., 2016).

| Answer Type | Percentage ( %) |
|---|---|
| Date | 8.9 % |
| Other Numeric | 10.9 % |
| Person | 12.9 % |
| Location | 4.4 % |
| Other Entity | 15.3 % |
| Common Noun Phrase | 31.8 % |
| Adjective Phrase | 3.9 % |
| Verb Phrase | 5.5 % |
| Clause | 3.7 % |
| Other | 2.7 % |

Table 2.2: Diversity of questions in *SQuAD*.

To evaluate the systems two different metrics are used over a clean answer (removing punctuation, articles, etc):

- Exact match (*EM*):
  If the answer matches exactly with any of the ground-truth answers.

- *f1-score* (*F1*):
  Considering the answer and the ground-truth answers as a bag of words, the *f1-score* is computed between them to measure the average overlap.

Authors baseline(Rajpurkar et al., 2016) (consisted on window sliding) got a 13.0 % in *EM* and 20 % in *F1* (both of them in test set). A *Logistic Regression* got 40.4 % and 51 % respectively and human performance gets 77 % and 86.8 %.

Rightnow, the top #1 benchmark for *SQuAD* is *LUKE*(Yamada et al., 2020) by Yamada et al. that gets a 90.2 % in *EM* and 95.4 % in *F1*.

But models trained only on this *dataset* have a problem. If the question is not answerable within the context they make an unreliable response anyway. To solve this *SQuAD* was updated to *SQuADv2*(Rajpurkar, Jia, y Liang, 2018) with more than 50.000 questions that are indeed not answerable. This way, models have to know when they can answer a question and when they can not.

With the same metrics the new human performance is 86.8 % in *EM* and 89.4 % in *F1*. Top #1 model is *SA-Net on Albert (ensemble)* and gets 90.724 % in *EM* and 93.011 % in *F1*.

### RACE

*RACE* (Large-scale ReAding Comprehension Dataset From Examinations)(Lai et al., 2017) is a *dataset* created by Lai et al. in 2017 for *Multiple-Choice QA*. It consists of near 28.000 passages and near 100.000 questions with 4 possible options each. These questions and passages were collected from English exams for middle and high school Chinese students in the age range between 12 to 18. This way, questions were made by English teachers to enhance and measure the English knowledge of the students, so they cover different topics and are designed to require a high level of reading comprehension and reasoning.

As *RACE* covers a wide range of ages (12-18), the authors splitted it into *RACE-M* and *RACE-H*. *RACE-M* denotes the passages and questions collected from the middle schools students (12-15 years old), meanwhile *RACE-H* denotes the ones collected from the high schools students (15-18 years old). Then bot of them were splitted into training set (90 %), development set (5 %) and test set (5 %). Table 2.3 shows the number of passages and questions in each with a total number of passages of 27.933 and 97.687 different questions.

| Dataset | *RACE-M* | | | *RACE-H* | | | *RACE* | | |
|---|---|---|---|---|---|---|---|---|---|
| Subset | Train | Dev | Test | Train | Dev | Test | Train | Dev | Test |
| # passages | 6.409 | 368 | 362 | 18.728 | 1.021 | 1.045 | 25.137 | 1.389 | 1.407 |
| # questions | 25.421 | 1.436 | 1.436 | 62.445 | 3.451 | 3.498 | 87.866 | 4.887 | 4.934 |

Table 2.3: Number of passages and questions in each *RACE* set.

As it can be seen the number of passages and questions is much larger in the *RACE-H* set, the same as the vocabulary size and the length of the passages, questions and options. Although the vocabulary size is large (136.629), authors note that as the passages and questions have been collected from English exams to Chinese students the vocabulary is not as complex as in other *datasets* where data is collected from news, *Wikipedia* or other sources.

To understand the reading comprehension and reasoning required by *RACE* authors analyzed the *dataset* "classifying" the questions into:

- Word matching: The question exactly matches a span in the article. The answer is self-evident.
- Paraphrasing: The question is entailed or paraphrased by exactly one sentence in the passage. The answer can be extracted within the sentence.
- Single-sentence reasoning: The answer could be inferred from a single sentence of the article by recognizing incomplete information or conceptual overlap.
- Multi-sentence reasoning: The answer must be inferred from synthesizing information distributed across multiple sentences.
- Insufficient/Ambiguous: The question has no answer or the answer is not unique based on the given passage.

And, more specifically into:

- Detail reasoning: to answer the question, the agent should be clear about the details of the pas-sage. The answer appears in the passage but it can-not be found by simply matching the question with the passage.
- Whole-picture reasoning: the agent needs to understand the whole picture of the story to obtain the correct answer.
- Passage summarization: The question requires the agent to select the best summarization of the passage among four candidate summarizations.
- Attitude analysis: The question asks about the opinions/attitudes of the author or a character in the story towards somebody or something
- World knowledge: Certain external knowledge is needed like simple arithmetic, geography, etc.

A complete analysis and different methods can be found in (Lai et al., 2017). In the article authors compare human performance with the results of different methods such as windows sliding or *Stanford AR*. The human performance is 95.4 % for *RACE-M* and 94.2 % for *RACE-H*. The top #1 right now is *Megatron-BERT (ensemble)*(Shoeybi et al., 2019) by Shoeybi et al. from *Nvidia*, that achieves an accuracy of 93.1 % for *RACE-H* and 90.0 % for *RACE-M*.

### QuAIL

*QuAIL* (Getting Closer to AI-complete Question Answering: A Set of Prerequisite Real Tasks)(Rogers et al., 2020) is a *dataset* created by Rogers et al. in 2020 for *Multiple-Choice QA*. In the article, authors say that although there are several *datasets* for

different reasoning tasks (such *QA*), most of them get "solved" almost immediately, and this is because of the poor diversity of the data, that has bias and spurious patterns that make the models to recognize these patterns more than to actually learn the language. To solve this, the authors have analyzed different *datasets* and have balanced different types of data to try to reduce the amount of bias and spouring patterns.

This has led the authors to find that, for instance, paraphrasing makes models to miss performance, even for *BERT* and other *LM*.

*QuAIL* is a multi-domain *dataset* with 200 texts for each of the 4 domains available. Questions about these texts have 3 different options and one *Not Enough Information (NEI)* question, that it's the correct option for unanswerable questions. The texts are from 4 different domains: CC license fiction, Voice of America news, blogs and user stories from Quora and each one has 18 question ($\tilde{1}$4k questions). The question are classified into the next types:

- Text-based questions: Those questions that the information needed to answer should be found in the text.
  - Reasoning about factual information in the text.
  - Temporal order of events.
  - Character identity.
- Word knowledge questions: Those questions that the answer rely on some kind of inference about characters and events, based on information in the text and world knowledge.
  - Causality.
  - Properties and qualities of characters.
  - Belief states.
  - Subsequent state after the narrated events.
  - Duration of events.
- Unanswerable questions.

To avoid some possible co-occurrence of words the authors performed paraphrasing in a total number of 556 questions of 30 fiction texts, aiming in particular to reduce lexical co-occurrences between the text and the words in the question and the correct answer. Results showed that all models suffered a significant drop in performance, specially for *BERT*.

The official benchmark[1] shows that only the authors have submitted their models, so it's not possible to compare models yet, but the top #1 of the leader-board is for the model *TML BERT Baseline* with a 53.4 % for all questions.

## 2.3. Explainability

Deep Learning systems are highly used among society and most of the companies nowadays. These technologies have revolutionized the global thinking of business, allowing to make efficient processes and to solve new tasks unthinkable until now like those involving language understanding. Undoubtedly *AI* systems offer huge benefits and enhance a large number of human activities, however, the use of these systems in some critical business processes like people hiring or credit granting are huge decisions and it is not clear how the model is making these decisions and what are the features in which the model is trusting in order to make the predictions. This way, the effectiveness and insights obtained by the models are limited due to the inability to explain the decisions taken by the models and the repercussion that those decisions may have on the users.

This fact has created many concerns about the risks that they introduce and has reduced the initial excitement about the transformation and improvements introduced by these systems. Concerns that have been raised include possible lack of algorithmic fairness (leading to discriminatory decisions), potential manipulation of users, potential lack of inclusiveness, infringement of consumer privacy, and related safety and cybersecurity risks. The use of artificial intelligence explainability (*XAI*) can solve these concerns and provide several business benefits such as:

- Explanations can help to identify problems in data and features so it can improve the model performance at the same time that can help to better decision making thanks to the additional information provided.
- Gives a sense of control and safety since the *AI* system owner knows in every moment how the behavior is and each decision can be subjected to pass through safety guidelines and alerts on its violation.
- One of the main advantages is the possibility to build trust around the model with the stakeholders who can interpret every decision made. Moreover it can monitor ethical issues and violation due to bias in data.
- Lastly, it can solve compliance issues related with accountability and regulation. (E.g. Adherence to *GDPR* regulation where "Right to Explain" is must-have for a system).

*XAI* will take more and more importance and a significant part of the effort from the research community and companies will be focused on creating interpretable models or

[1] http://text-machine.cs.uml.edu/lab2/projects/quail/

providing a set of tools that help to understand and explain the decisions taken by the *AI* models. The goal is to enable users to understand, trust model predictions while maintaining the learning performance.

### 2.3.1. XAI classification

Although explainability is a recent technique, there are already a lot of different techniques and systems to try to explain models and predictions. Even tough there a lot of techniques, all are based on the same concepts and, therefore, they can be classified depending on their nature in different aspects:

**Global Interpretation vs Local Interpretation** Among the different classifications that can be done in order to difference between all the *XAI* techniques, the most important is the type of the explanation. The are two different types of explanations: the explanation of how a model works and the explanation of why a model has given a specific output as a result.

This way, the *Global Interpreters* try to understand the whole model, how it works and what *features* are important for the model to explain the entire model's predictive reasoning. This is very interesting for some use cases, like for instance to understand if the model has some *bias*. On the other hand, the *Local Interpreters* try to explain a unique prediction of the model, by looking at a local subregion in the feature space around the prediction, trying to understand the model decision based on that local region.

**Model-specific vs model-agnostic** Other interesting classification of the *XAI* techniques available is to know which ones can be applied to all models or which ones can only be applied to a specific model type. If, for instance, a company has different models being used (e.g. Support Vector Machines, Random Forests and Neural Networks) its more interesting for the company to use the same tool for all models rather than one different technique for each one. Model-specific tools are those one that can be only applied to a specific model type (for instance, those tools that study the *attention* mechanism can only be applied to language models that use *attention*). Model-agnostic, on the other hand, are those tools that can explain every model, no matter its type.

**Intrinsic vs post hoc** Simple machine learning models can be explained themselves, this interpretability is considered as intrinsic e.g. rules-based, linear models, parametric models or tree based models. In these types of models the explanation arise itself with the output of the model, and there is no need of post processing to generate an explanation. However, complex models such as Neural Networks require some additional processing to be performed after the predictions are made in order to get the explanation.

### 2.3.2. XAI Techniques

Although there are a lot of different techniques available to explain models or predictions, there are some techniques that are the most frequent in the literature. Most of the available tools for *XAI* use these techniques to give the explanation.

**Feature Importance** The model generates the output by performing operations over the inputs, this is, the *features*. Therefore, to understand the importance of the *features* can help to understand the model behavior and to explain it.

**Surrogate Model** If the type of the model is not easily interpretable (such as Neural Networks) or if it is not known, a common way to explain it is to build a second model easily explainable, that will try to behave exactly the way that the model to explain does. This way, by explaining the second model the first one can be explained.

**Example-Driven** This technique is based on search for a similar labeled instance to the input, that will be used as an explanation. These similar instances are obtained from labeled data and they can be gotten by studying semantic similarity, or nearest neighbors in the *feature* space.

**Provenance** In some systems the prediction process involves a series a of reasoning steps, in this cases, the prediction steps can be illustrated to be used as explanation.

**Induction** Some way to explain the model functioning is to generate human-readable representations, such as rules or decision trees that are induced as explanations.

**Anchors** Models use to have some bias in the training data (and therefore in themselves). There are also some inputs that condition an specific output in a way that, if input $x$ is present, the output is always going to be $o$. This is known as an *anchor* and this technique looks for anchors to help to understand a local prediction.

These techniques are all based in solving a few operation that enable explainability. Although some of them depend on the type of model being used (*model-specific*, other are applied to any kind of architecture (*model-agnostic*) such as the *Surrogate Model* technique, that usually is trained with *perturbations of the input*(Ribeiro, Singh, y Guestrin, 2016) mades by generating random instances modifiying the input $x$. Another technique that can be used with different types of models (althoug it is mostly used with *NN*) is the *first-derivative saliency*, which consists on "estimate the contribution of input $i$ towards output $o$ by compuing the artial derivative of $o$ with respect to $i$".(Danilevsky et al., 2020)

On the other hand, the *model-specific* techniques use to involve some particularities of the model. For instance, if using *RNN*, and more specifically *LSTM*, the output of the *LSTM* cells and the output of the "gates" within them can be used to explain the outputs of the model (*Feature Importance* technique, (Ghaeini, Fern, y Tadepalli, 2018)). More recently, if using *LM* based on *transformers* and/or *attention* layers, the information provided by these layers can be used to explain the behaviour of the model, and because of this there are plenty of works that made use of this startegy, (Luo et al., 2018), (Xie et al., 2017), (Serrano y Smith, 2019), (Jain y Wallace, 2019).

### 2.3.3. XAI Visualization

The main point of *XAI* is to help users and developers to understand the output of a model, why that prediction, why not another one or what can be done to improve the model performance. So, a really important point is to understand the *XAI* techniques outputs, because if the user does not understand the explanation it's not possible that he understands the model behavior.

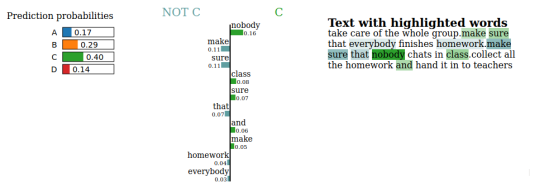
Because of this, the *XAI* visualization is as important as the techniques used to get the explanation and, therefore, there are a lot of different ways of visualize explanations, depending on the type of the technique used, the model, the kind of the problem (*NLP*, tabular data, computer vision, classification, regression, etc).

### 2.3.4. XAI Evaluation

Due to the young age of the *XAI* techniques and to the difficult of the problem being treated, there is no a standard way to evaluate the *XAI* systems and techniques. Most of the papers in the literature that present solutions (Xie et al., 2017) don't evaluate the systems and just present a simple discussion and explanation of the *XAI* results based on what the authors think, using only selected examples and not evaluating the system at its whole.

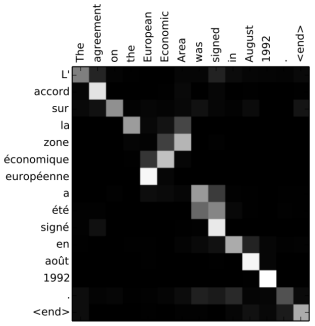
Some approaches try to solve this by generating ground truth evaluations to quantify the performance of the *XAI* techniques, by using well known evaluation metrics such as *Precission, Recall and F1-Score*(Carton, Mei, y Resnick, 2018) or *BLEU*(Ling et al., 2017), (Rajani et al., 2019). Although is an interesting way to evaluate the systems, usually there is no only a valid *XAI* explanation, but different explanations may be valid. Because of this, the ground truth acquisition is critic to get a valid evaluation.

Other approaches try to solve the problem of multiple valid explanations by using human evaluation. This way, humans can evaluate the quality of the explanations without having to generate ground truth or to measure the similarity of the explanations. This is

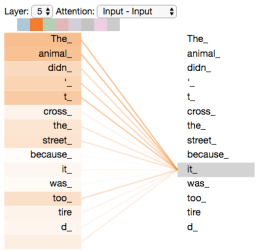



(a) Visualization of *LIME*.


(b) Visualization of *SHAP*.




(c) Attention Visualization




(d) *BertVIZ* Visualization.

Figure 2.16: *XAI* Visualization examples.

also interesting, because *XAI* systems try to explain the behavior of a machine learning model, and this is usually a black box humans don't understand and, therefore, can not generate a real ground truth.

Also, as said before, one important step of the *XAI* pipeline is the visualization, and a *XAI* system that can explain the result better to the user is more effective than another one that does not gives a good visualization for explanations understanding, even if this last one has a greater explanation of the model behaviour. This way, the explanations can be evaluated by its understanding of the model prediction and by the human comprehension of the explanations. Lertvittayakumjorn et al. (Lertvittayakumjorn y Toni, 2019) present three different ways of evaluate the *XAI* systems based on the different goals of these systems: To reveal accurately the model behavior; to explain the model predictions; to assist humans in future investigations about uncertain predictions.

As *XAI* systems effectiveness depends on the human understanding, is important to have multiple evaluators to deal with different responses and subjectivity, like did Sydorova et al. (Sydorova, Poerner, y Roth, 2019), where 25 humans evaluated and compared explanations of different techniques.

One way mostly used to evaluate the explanations of *attention* layers is to set to zero the maximal entry of the layer to see if the prediction is altered (Serrano y Smith, 2019). This way, if the result is altered by turning off the dominant weights, this would mean that the weights are indeed explaining the result. Same can be applied to other methods, for instance the replacement of the most important words for a specific prediction could change the result if that are actually important words for the prediction.

## 2.4. Explainabity in NLP

The same that has happened with other types of *AI* problems, such as time series regression or in computer vision, has happened to *NLP*, this is, the usage of *Deep Learning* to improve the quality and the results of the models and, with it, the lack of explainability.

At the beginning, *NLP* models where mainly based in *white box* techniques, such as *Hidden Markov Models (HMM)*, *Decision Trees* or *Logistic Regressions* which are inherently explainable, but with the arrival and popularity of *Deep Learning* models and *word embeddings* as features, it has become almost impossible to understand why a model gives an output instead of another one.

Because of this, many researchers and companies have focused their research in the explainability, and its application to the *NLP* field, developing systems that are able to explain the model behavior and which features (or word, in case of *NLP*) are more important for the model prediction. But, even tough explainability in *Deep Learning* models is difficult *per se*, and more in *NLP* problems, sometimes it gets even more difficult depending on the problem type.

Some *NLP* problems are more complex than others. For instance, text classification problems are less complex than text summarization (depending, of course, on the use case). This is because sometimes there are some words that only appears in some classes, so if that words appear in the text it can be classified without understanding the language. Same happens with simple named entity recognition, where some features like prepositions before the entity, the uppercase and so on help a lot to recognize a named entity. But some problems, like text summarization and question answering are more complex tasks in which the model has to understand the language and the text to be able to solve the task.

To understand the language is something really difficult and only humans can do, a lot philosophers say that is language what separates humans from animals actually, and, of course, its complex for machines also. Meanwhile machines are grater than humans while treating with numbers, they can not understand languages as well as humans

do, due to the complexity of the task, that involves not only to understand words and grammar, but also to have an understanding about the world, science, politics, history, etc. This makes some tasks really difficult and, although some models have achieved a really good result in some of this tasks, it is not clear if it is because these models understand the language and the task or if it is due to an over-fitting. This can be seen in text summarization, where models trained over scientific papers or over news can not summarize other texts with the same performance, giving as output short sentences (in case of models trained over news), that are not always a good summary.

This is why *XAI* is difficult in *NLP* tasks, but also this makes *XAI* important, because if users don't know why the model has given an output, they can not trust in it. However, researcher have been working in explain *NLP* models for the last 5 years, developing really good approaches.

In 2015 Voskarides et al.(Voskarides et al., 2015) use a two-step approach for automatically explaining relationships between entity pairs, selecting sentences that explain the triplets and using machine learning to rank them according to how they explain the relationship between the entities. Other researchers has also pointed out the utility of applying knowledge graph to deep learning models to increase not only the performance but also the explainability of the models (PALMONARI y MINERVINI, 2020), (Lecue, 2020). In 2019, Moon et al. presented *OpenDialKG*(Moon et al., 2019), that use knowledge graph to create a dialogue and reasoning system, that also generates a walk path, providing a way to explain conversational reasoning.

But, although there are a few papers explaining others types of systems such as machine learning algorithms, (Pappas y Popescu-Belis, 2014), most of the papers with model-specific techniques are focused on the neural networks due to the difficulty to explain them and their great performance. Gupta et al. proposed a technique named as *Layer-wIse-Semantic-Accumulation (LISA)*, (Gupta y Schütze, 2018) for explaining RNN, detecting patterns on while decision making. Poerner et al. developed a method called *LIMSSE*, an adaption of *LIME* to NLP, that achieves good performance in NLP and DNN. Something like that did Wallace et al. (Wallace, Feng, y Boyd-Graber, 2018), using *Deep k-Nearest Neighbors*(Papernot y McDaniel, 2018) for text classification. As they say in the article, the confidence of neural networks is not a robust measure of model uncertainty, making local interpretation methods less robust in these cases.

In 2016, Lei et al. proposed a framework in which they used and encoder and a generator to solve the problem (Lei, Barzilay, y Jaakkola, 2016). The idea is to generate *rationales*, that are pieces of input text as justifications, that once they are feed to the encoder the result of the prediction is the same but that they can help to justify the decision.

Alvarez-Melis et al. proposed a framework to explain black-box models based on input perturbation that returns as explanation a set of input-output tokens that are related to each other inside the model (Alvarez-Melis y Jaakkola, 2017). Input perturbation is one of the techniques most used in *XAI*, but Alvarez-Melis et al. proposed a novel way of generating these perturbations, by using a variational autoencoder to generate similar sentences and, therefore, controlled perturbations.

Another interesting approach is that proposed by Rajani et al. in (Rajani et al., 2019), in which they develop a system named Commonsense Auto-Generated Explanation (*CAGE*) to generate explanations in natural language, by training ~~a model (*GPT-2*) on a~~ new *dataset* called Common Sense Explanations (*CoS-E*).

Survey of XA
QA

## 2.5. Explainability tools

During the past years some tools have been developed to explain (or at least help to understand) machine learning models. Some of these tools can be applied to *LM* and some of them can be applied to *QA* and *Multiple Choice* problems. Next ones are some of the most famous explainability tools.

### 2.5.1. LIME

*LIME* (Local Interpretations Model-Agnostic Explanations) is a method developed by *Marco Tulio Ribeiro* to explain any black box classifier, with two or more classes.

It's able to explain local predictions of almost any machine learning problem, no matter is a computer vision problem, a *NLP* problem or a problem based on tabular data. As it's model-agnostic it's also able to explain every single algorithm, as it considers the model as a black box.

The implementation of this method is available in the *GitHub* of the author [2] with a *BSD-2 Clause license*, so everyone can use the method. This has made this method very famous, being very used by the community. The usage of the tool is very easy. Users have to implement a function that takes as input a *NumPy* array with instances and outputs the result of the prediction, with the probability for each class.

**Advantages:**

- Model-agnostic: It can explain any machine/deep learning model.
- Easy visualization.
- Fast computation.

[2] https://github.com/marcotcr/lime

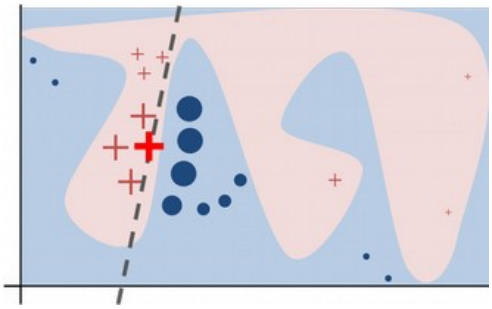


Figure 2.17: Functioning of *LIME*.

**Disadvantages:**

- Local: *LIME* is just local explainer, it cannot explain the entire model behavior.
- Randomness: score can be different every time as it generates samples randomly so it lacks of consistency.

**Method Description** This algorithm takes the assumption that is possible to fit a simple interpretable model around a single observation that will mimic how the global model behaves at that locality. This is what makes *LIME* a model-agnostic method, as it will explain the new model created, which is an interpretable model (*RIDGE*), regardless the type of the model to be explained.

It works by perturbing the input and seeing how the predictions change for each perturbed input. It first generates a neighborhood data by randomly hiding features (words in this case) from the text instance. Then, it generates an explanation by approximating the underlying model by an interpretable one learned on the perturbations of the original instance (e.g., removing words).

This way, it learns locally weighted linear models on this neighborhood data to explain each of the classes in an interpretable way and indicate the impact of word existence.

In the Figure 2.17, the blue/pink background corresponds to the non-linear model being used in the predictions, that has two labels, crosses and circles. While explaining a local predictions (the red cross), *LIME* generates random samples and weights them according to the distance to the prediction being explained (the size of the crosses/circles). The dashes line correspond to the linear model created by *LIME* and its the one being actually explained by *LIME*.

**Application to Language Models** As *LIME* is a model-agnostic method, it's able to explain every type of model, included Language Models. *LIME* is going to consider the original Language Model a black box, creating an additional model, fitted with perturbations of the input. This way, is able to interpret local predictions of Language

Models. The perturbations will be made over the instance of the local prediction by, for instance, removing words.

Language Models are based on the prediction of the next word (or sequence), which is more like a regression problem. Even when *LIME* is also able to explain regression problems, is more focused on the classification problems. Nevertheless, the most of the Language Models used nowadays aren't used to predict the next sequence, but to do complex NLP tasks, such as text classification, Named Entity Recognition or Question Answering, and this is done by adding additional layers at the top of the model. This layers will be different depending on the kind of the problem, but most of them can be treated as classification problems. For instance, the Named Entity Recognition problem consists on classify each token as a named entity or not. This way, *LIME* could be used not only to explain the Language Models themselves, but also the complex model built on top of them, such as *BERT*.

**Application to Question Answering** Question Answering models can be classified into two different types. The ones that try to extract the answer directly from the text, given as the result the tokens inside the text that answer the question (with *datasets* such as *SQuAD*) and the ones that try to select the correct answer among the possible answers given, which are known also as multiple choice problems (with *datasets* such as *RACE*).

Depending on the QA type (extractive or multiple-choice) the way to proceed is different, but both of them can be treated as classification problems. In the extractive ones, span classification layer is added on top on the base model (such as *BERT*), and in the multiple-choice ones a classification layer is added on top of the pooled output to classify which one is the correct answer.

This way, considering the QA problem as a classification one, is easy to adapt *LIME* to use it in this kind of tasks. In these cases, *LIME* can take as input the context, the question, the options or any combination possible between them, explaining for instance which words are important of every single option.

### Attention Heatmap

One of the most common ways to visualize the *Attention* values is with a heatmap of the values, Fig 2.16c.

The *Attention* method relates each *token* with the others and this methods plots those relationships as a heatmap. This way, it shows the *Attention* value of each *token* in y-axis attending to each *token* in x-axis. But in the *LM* it is common to have multiple

heads, so there are two options, to plot a heatmap per each head or to plot a heatmap with an output of all heads at once, that can be the average of the values, the sum, etc.

To visualize the *Attention* heatmap is an easy way to see how tokens relate with each others, which may help to understand what is going on inside the model. Although after several experiments (not only of this method but others, as will be seen after) it has been seen that the *Attention* values is not really useful to explain why the model has predict an specific output instead of another one, it can be used, as seen, to detect patterns, useless heads and so on.

As an explainability method it is true that it is not as helpful as other methods that are indeed trying to explain the model output. But it can be used to detect if the model is under-fitted (or over-fitted), which may explain a wrong prediction. But it does not help to explain a right prediction.

**Advantages:**

- It allows to visualize the *attention* values easily.
- Heatmaps are easily-interpretable visualizations than can make humans to understand model insights in an easy way.

**Disadvantages:**

- It just shows the way *tokens* relate with each other in 2D and between two sentences..

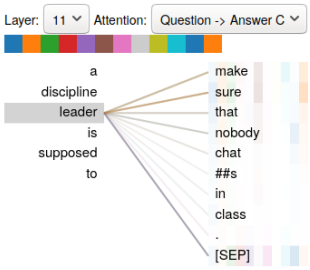
**BertViz**

*BertViz*, presented by Jesse Vig in 2019 (Vig, 2019), is one of the most famous library for visualizing *Attention* layers. It presents 3 different types of visualization, including:

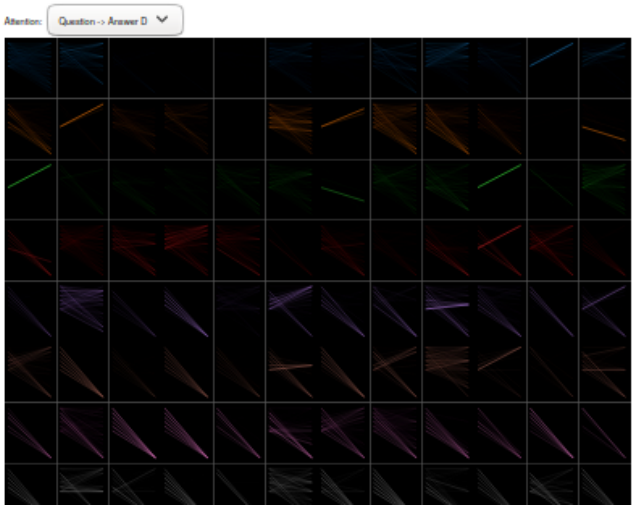
- Head view: Visualize the *attention* patterns in a single layer, Fig 2.18a.
- Model view: Visualize the *attention* patterns in every layer at once, Fig 2.18b.
- Neuron view: Visualize the *query* and *key* vectors of a single neuron to show how *attention* works, Fig 2.18c.

It also supports all models from the *transformers*[3] library of *Huggingface*[4], which is the most used library for *transformers* usage and development. This way, it is easy to use the tool and visualize the models used.

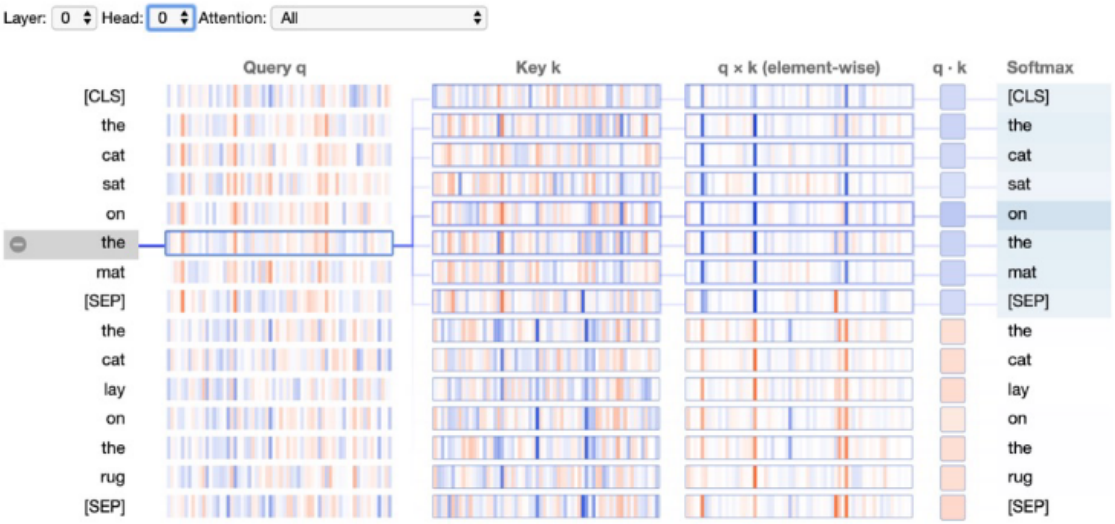
[3] https://github.com/huggingface/transformers
[4] https://huggingface.co/




(a) Head view of *BertViz*.



(b) Model view of *BertViz*.




(c) Neuron view of *BertViz*.

Figure 2.18: *BertViz* examples.

The implementation of this method is available in the *GitHub* of the author [5] with a *Apache 2.0 license*, so everyone can use the method even for commercial use.

The usage of the tool is very easy, but depending on the problem. It supports only the visualization between two sentences, and when the input is too large it fails or takes too long to show the result. However, for most of the problems this is enough to see the model behavior.

**Advantages:**

- It allows to visualize the *attention* in different ways, which makes possible to deep

[5] https://github.com/jessevig/bertviz

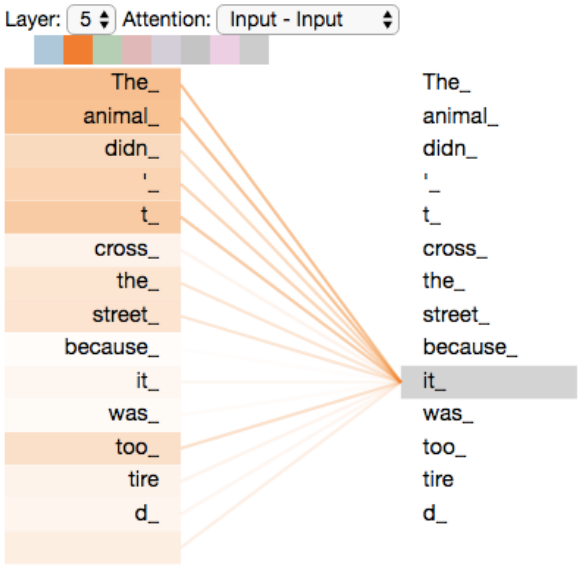



Figure 2.19: Example of *BertViz*.

into the model and see how it works.

- It supports all models from the *transformers* library, which makes it an easy tool for visualizing the *attention*.
- The interactive visualization provided (made with *Javascript*) makes it easy and user-friendly.

**Disadvantages:**

- It only works with two sentences, which is not enough sometimes.
- It only works with short inputs.
- The Neuron View only supports *BERT*, *GPT-2*, and *RoBERTa* models.

**Method Description** As *BertViz* is only a visualization tool, it does not process anything (like e.g. *LIME*, which has to train a surrogate model).

*BertViz* takes the *Attention* weights produced by the model in the *forward* and shows them in pretty and useful way, making simple to understand them.

Figure 2.19 is an example of *BertViz*, in the *Head View*. It shows how *tokens* at left attend to *tokens* at right in one layer (and $n$ heads). The tool provide 5 different options:

- All: Shows how all *tokens* attend to every *token*.
- Sentence A $\rightarrow$ Sentence A: Shows how *tokens* in Sentence A attend to *tokens* in Sentence A.

- Sentence A → Sentence B: Shows how *tokens* in Sentence A attend to *tokens* in Sentence B.
- Sentence B → Sentence B: Shows how *tokens* in Sentence B attend to *tokens* in Sentence B.
- Sentence B → Sentence A: Shows how *tokens* in Sentence B attend to *tokens* in Sentence A.

**Application to Question Answering** By default, *BertViz* does not support *QA* tasks, in the way that it just shows the relation between two sentences. However, it can be considered as Sentence A the question and each option as Sentence B.

This way, some modifications have been developed to make it easier for the user. In these modifications, developed for *Multiple-Choice* problems, the user just has to input the example (including the text, the question and the options) and the modified *BertViz* shows 8 different options:

- Question → Answer A: Shows how *tokens* in the Question attend to *tokens* in the Answer A.
- Question → Answer B: Shows how *tokens* in the Question attend to *tokens* in the Answer B.
- Question → Answer C: Shows how *tokens* in the Question attend to *tokens* in the Answer C.
- Question → Answer D: Shows how *tokens* in the Question attend to *tokens* in the Answer D.
- Answer A → Question: Shows how *tokens* in the Answer A attend to *tokens* in the Question.
- Answer B → Question: Shows how *tokens* in the Answer B attend to *tokens* in the Question.
- Answer C → Question: Shows how *tokens* in the Answer C attend to *tokens* in the Question.
- Answer D → Question: Shows how *tokens* in the Answer D attend to *tokens* in the Question.

Another modifications were made to show relations regarding *tokens* in the context, but the input in that case was too long for *BertViz*, so it was not possible to show it due to hardware limitations.

### 2.5.2. Intrinsic GPT-3

*GPT-3* has demonstrated to be a very powerful model that is able to chat, to answer question, to summarize and even to code functions based on a description of the functionality desired. To do this and to force it to act as expected, it's enough to feed it a few examples with the result expected and then *GPT-3* will generate the output.

Therefore, if the user feed it with a few questions (starting by a special *token Q:*) with their answers (starting also by a special *token A:*), when the user inputs only a question, the model will generate the answer.

Although *GPT-3* has not been trained for any task specifically, it has demonstrated to be able to solve them by an unsupervised zero-shot learning, so it may be able to give an explanation of the answer.

# Chapter 3

# Methodology

The proposal of this work is to review the state of the art of *XAI* of *LM* applied to multiple choice problems. In this chapter the methodology followed is explained.

## 3.1. Description

In order to test the different *XAI* tools, a language model for a multiple choice problem is needed. So the first step covered is to train a *LM* to be used. Then the different *XAI* tools to be tested are going to be adapted to multiple choice problems.

## 3.2. Model Training

As said before, a language model was trained for a multiple choice problem. Although it is not the best choice for a real problem (nor the one that has the best performance nor the fastest model), *BERT* has been chosen, because it is the most famous language model.

Therefore, a *BERT* model is going to be trained over one of the multiple choice *datasets* seen in Section 2.2.2. The *dataset* selected was *RACE*, taken from the Huggingface *datasets* library. [1].

The implementation of *BERT* used was the one available at the Huggingface *transformers* library. [2] Due to hardware limitations the *BERT base* version was used, and as it is not as powerful as the largest version, the *RACE-middle* version of the *dataset* was used. The accuracy obtained after 3 epochs was of 0,641. Although it could be improved, the goal of this work is not to train a good model for the multiple choice problem but to explain the model decisions, both right and wrong. The code to train the model can be found in:

[1] https://huggingface.co/docs/datasets/
[2] https://huggingface.co/transformers/

- GitHub: https://github.com/marmg/TFM/blob/main/BERT-RACE.ipynb
- Google Collab: https://colab.research.google.com/drive/1RexG8T8Rz8QvVkXOH-K5VaXpuSY-mAEJ?usp=sharing

### 3.2.1. Explainability model-agnostic

As seen in **??**, there are different types of *XAI* tools. In this work both model-agnostic and model-specific tools are going to be tested. The probably most famous *XAI* tool is *LIME*, and therefore is going to be tested.

#### LIME

In order to use *LIME* with *BERT* for Multiple-Choice (the same as with any other model), a function has to be developed that takes as input a list of perturbed instances, makes the prediction and return the probability for each class.

Even when is not the efficient, the easiest and most common way to fit the model is by repeating the article, joined to the question and one of the options, for instance while predicting in *RACE* dataset, the result would be four instances with the same article and the same question, but with a different option each.

So, for developing the function used by *LIME*, this four instances have to be created inside it with the perturbations made by the method. Depending on the experiment, this instances will be different because the perturbations will be just the question, the article, the options and so on, and will be concatenated with the original parts that are not being perturbed.

Then, the instances are converted to the features to make the prediction. The result is transformed with a *Softmax* function and finally is returned to *LIME*, which analyzes the outputs related to the perturbations made, given as the result a score based on the importance of the word for the prediction.

*LIME* works by perturbing the instance to measure the importance of the words, that is enough for most of the *NLP* tasks such as Text Classification. But in Multiple-Choice task there are different parts to take into account, the article or context, the question and the options. Thus, four experiments are going to be made to explore each of the parts.

1. 1st experiment: The question is going to be perturbed.
2. 2nd experiment: The options are going to be perturbed.
3. 3rd experiment: The article is going to be perturbed.

4. 4th experiment: The question and the options are going to be perturbed.

With this, not only the most important words are expected to be detected, but also which part is more important for the model.

As *LIME* takes as input the instance to perturb and the prediction function, different functions have to be developed in order to make the experiments. One of them will take as input the question and will generate the features of the example by using the original article and options.Other will take as input the options and once again will generate the features of the example by using the original article and, in this case, the original question. The same with the article and with the question and options together.

When there are more than one instance (like in the options), they are going to be concatenated with a special character and the function will split it to generate the different examples.

### 3.2.2. Explainability model-specific

Among the different model-specific options available to study the *LM*, the most famous and useful is to look at the *Attention* weights. *Attention* is used to vectorize the words generating contextual embeddings, taking into account how words relate themselves and what words are in the text to understand the real meaning of the words.

This way, to look at the *Attention* has proved to be useful to understand the model behaviour and to be able to prune heads of the *LM*, reducing the size of the model, the time or the hardware needed and, therefore, several tools have been developed to study and to look at the *Attention* layers.

#### Attention Heatmap

The *attention* heatmap is a correlation plot that show hot *tokens* in one sentence attend to *tokens* in other (or the same) sentence. As this only depends on the *tokens* and values that are shown, it is possible to plot a heatmap of *Multiple Choice* and *QA* problems, showing how *tokens* at question attend to *tokens* at options.

#### BertViz

Although *BertViz* does not support *QA* or *Multiple Choice* problems by default, it is possible to apply it to these types of problems by modifying the source code, which is open and available in GitHub[3].

[3] https://github.com/jessevig/bertviz

*BertViz* takes as input the *attention* weights and use them to draw the correlation between *tokens* based on these values. Thus, if the *attention* values of a *Multiple Choice* problem is taken as input and it is parsed it is possible to adapt the tool for these problems. The modifications done format the *attention* values to show how *tokens* at question attend to *tokens* at each of the options and vice versa.

### 3.2.3. Intrinsic GPT-3

Following the idea of Rajani et al. (Rajani et al., 2019) and trying to take advantage of the power of *GPT-3*, one test was developed in order to see if *GPT-3* was able to explain itself.

To force *GPT-3* to generate the explanations (or any expected result), first a few examples of what is exactly desired have to be feed to the model in order to show it what is expected. In this case the model is going to be feed with some examples following the next structure:

- Context
- Question
- Options: Options formatted with an identifier (A-D).
- Answer: Identifier of the option chosen.
- Explanation: Explanation of the answer

In Appendix A the examples used are shown.

# Chapter 4

# Results

Before going deeper into the results, the experiments and examples used to test the tools are being introduced. As tools are different depending to its nature (model agnostic vs model specific), different experiments have to be done.

Besides this, examples used are going to be the same to be able to compare the results. 3 examples of the *RACE test set* have been selected and modified to test the tools. The original examples selected and the modifications are listed below. The part of the article where the answer is appears in bold.

**Example 1**

**Passage 4.1: Article of Example 1**

Take a class at Dulangkou School, and you'll see lots of things different from other schools, You can see the desks are not in rows and students sit in groups. They put their desks together so they're facing each other. How can they see the blackboard? There are three blackboards on the three walls of the classroom!
The school calls the new way of learning "Tuantuanzuo", meaning sitting in groups. Wei Liying, a Junior 3 teacher, said it was to give students more chances to communicate.
Each group has five or six students, according to Wei, and they play different roles .There is a team leader who takes care of the whole group. There is a "study leader" who makes sure that everyone finishes their homework. **And there is a discipline leader who makes sure that nobody chats in class**.
Wang Lin is a team leader. The 15-year-old said that having to deal with so many things was tiring.
"I just looked after my own business before,"said Wang. "But now I have to think about my five group members."
But Wang has got used to it and can see the benefits now.
"I used to speak too little. But being a team leader means you have to talk a lot.

> You could even call me an excellent speaker today."
> Zhang Qi, 16, was weak in English. She used to get about 70 in English tests. But in a recent test, Zhang got a grade of more than 80.
> "I rarely asked others when I had problems with my English tests. But now I can ask the team leader or study leader. They are really helpful."

Question: "*A discipline leader is supposed to _ .*"
Options:

- A: take care of the whole group.
- B: make sure that everybody finishes work
- C: make sure that nobody chats in class
- D: collect all the homework and hand it in to teachers

Correct Answer: C.
Prediction: C.
• **Modification** *a*:
Description of the modification: Question paraphrased.
New question: *What is a discipline leader supposed to?*
Prediction: A.
• **Modification** *b*:
Description of the modification: Question and options paraphrased.
New question: *What is a discipline leader?*
New options:

- A: A person supposed to take care of the whole group
- B: A person supposed to make sure that everybody finishes work
- C: A person supposed to make sure that nobody chats in class
- D: A person supposed to collect all the homework and hand it in to teachers

Prediction: B.
• **Modification** *c*:
Description of the modification: Question paraphrased and changed with a synonym and options paraphrased.
New question: *What is an orderliness leader?*
New options:

- A: A person supposed to take care of the whole group
- B: A person supposed to make sure that everybody finishes work
- C: A person supposed to make sure that nobody chats in class

- D: A person supposed to collect all the homework and hand it in to teachers

Prediction: C.

**Example 2**

**Passage 4.2**

traveler came out of the airport. There were a lot of taxis. He asked every taxi driver about his name. Then he took the third one. It cost 5 dollars from the airport to the hotel. "How much does it cost for the whole day?" The man asked. "100 dollars," said the taxi driver. This was very dear, but the man said it was OK.
The taxi driver took the man everywhere. He showed him all the parks and museums in the city. In the evening they went back to the hotel. The traveler gave the taxi driver 100 dollars and said, "What about tomorrow?" The taxi driver looked at the man and said, "Tomorrow is another 100 dollars." And the man said, "That's OK! See you tomorrow." The taxi driver was very pleased.
The next day the taxi driver took the traveler everywhere again. They visited all the parks and museums again. And in the evening they went back to the hotel. The man gave the taxi driver 100 dollars again and said, "I'm going home tomorrow." The driver was sorry because he liked the traveler and 100 dollars a day was a lot of money. "So you are going home. Where do you come from?" He asked. "I come from New York." "New York," the taxi driver said, "I have a sister in New York. Her name is Susan. Do you know her?" "Of course I know her. She gave me 200 dollars for you!"

Question: "*Where did the traveler come from?*"
Options:

- A: England
- B: America
- C: Canada
- D: France

Correct answer: B.
Prediction: A
• **Modification** *a*:
Description of the modification: Question paraphrased.
New question: *The traveler came from _ .*
Prediction: C.
• **Modification** *b*:
Description of the modification: Question and options paraphrased.

New question: *The traveler _ .*
New options:

- A: came from England
- B: came from America
- C: came from Canada
- D: came from France

Prediction: C.
• **Modification *c*:**
Description of the modification: Question paraphrased and changed with a synonym and options paraphrased.
New question: *The visitor _ .*
New options:

- A: came from England
- B: came from America
- C: came from Canada
- D: came from France

Prediction: C.

## 4.1. Model-agnostic experiments

Most of the model agnostic tools are based on instance perturbation, to train an underlying model easily interpretable that emulates the *LM* results.

Four different experiments have been developed to test the model agnostic tools and to try to understand which parts are more important for the model in order to make its prediction.

1. The first one was based on perturb just the question.
   The first idea was to try to explain which words of the question are important for the prediction, because at the end the model has to take as input the question and look for the answer. This way, the question the user asks to the model will determine the result of the prediction, and to know which words of the question are important is a good way to know how to make good questions.

2. In the second one, the options given to the model were perturbed.
   After the question, the most important part on the multiple-choice problems is the set of options given to the model, because it has to choose between some of this options.

3. The third one consisted on perturb the article.
   After all, the model has to look for the answer in the article, so it is important to that the model understands it. Thus, in the third experiment the article was perturbed.

4. In the final one, both the options and the question were perturbed.
   As sometimes question and options are related, sometimes even the options are the continuation of the question, it is also important to test the tools perturbing both options and question.

The main idea behind these experiments is to check if the tools are able to detect which parts are the most important for the model (the article, the question or the options), and this way know if the tools are useful to understand the result and if the result is legit or not. Besides this, to look for the words that are important for the prediction result is also expected from the experiments.

### 4.1.1. LIME

As seen before, *LIME* is a model agnostic tool based on instance perturbation. Thus, the first experiment consisted on make *LIME* perturb just the question and see what words are important for the model. This was tested with different samples of the test set (see **??**) and the results showed that *LIME* is able to detect the words that are indeed important for the model, as can be seen in Figure 4.1.

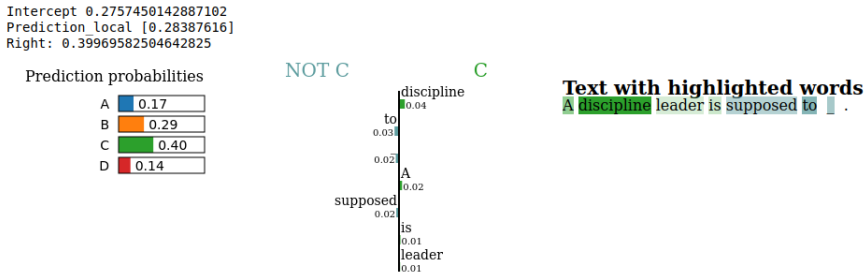



Figure 4.1: *LIME* result for the first experiment with original example.

In that example, the real label was $C$, and *LIME* was able to detect that the important word was "discipline", so if this word is changed the result will vary, but if the rest of the question is changed, the result shouldn't change too much. So the next steps were to check if that idea was true. In order to do this some changes were made in both the question and the options to see if the result was the same.

Following with the experiments planned before, the first modification was made in order to see if the question form is important for the model. As the important word, "discipline", has not being removed, the model shouldn't have any problem to get the

correct answer. Nevertheless, the model failed by selecting the answer *A* instead of the correct answer in this case.

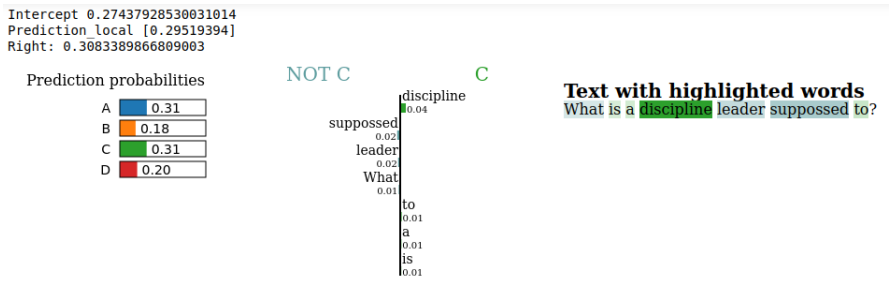



Figure 4.2: *LIME* result for the first experiment with the first change on the question.

But, attending to the probabilities of the prediction, it can be seen that both *A* and *C* options have almost the same probability. And, according to *LIME*, "discipline" is still important for the *C* option.

The next change was made to test if the question form is really important for the model. The change was made trying to do not change the meaning of the original ones, but adapting them to the answer expected by a "What is..." question. The result in this case was that the model also failed, but in this case it fails by choosing "B" as the correct answer.

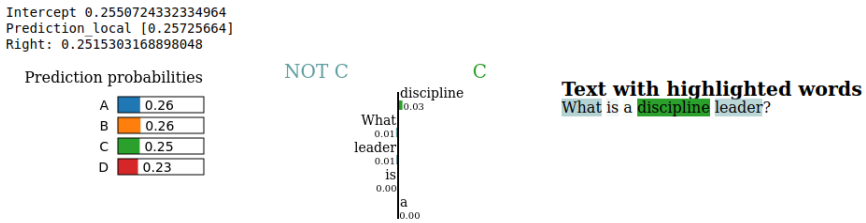



Figure 4.3: *LIME* result for the first experiment with the second change on the question.

To see if the question form was indeed so important and to see if the "discipline" word was important or not, a second change was made. This change consisted on replacing the "discipline" word with a synonym, "orderliness". This is interesting because in the original text the word "discipline" appears with the correct answer, but the word "orderliness" don't appear at all, so if the model is able to choose the correct answer could be due to the understanding of the model of the language, knowing the synonyms.

So the changes made before by paraphrasing the question were made now, but replacing "discipline" with "orderliness". The results were interesting because, in this case, the model chose the correct answer in both cases, something that didn't happened while preserving the original "most important" word.

As said before this could be because of the ability of the model to understand the language, but this was just an idea until *LIME* was used to check the importance of the word to see that the word "orderliness" was the most important word of the new question. The important thing here is that the importance score of "orderliness" is bigger in this case than the importance score of "discipline" in the other questions, what could explain the fact that in this case the model has been able to find the correct answer.

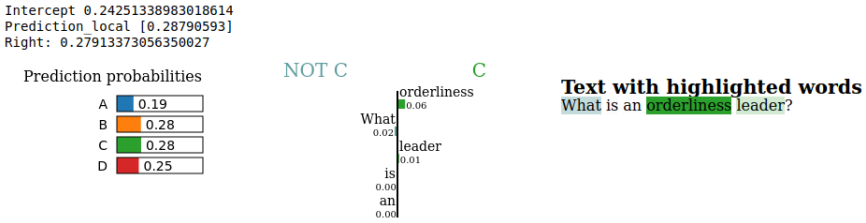



Figure 4.4: *LIME* result for the first experiment with the third change on the question.

Taking this into account, it is different the way to understand the model if it is a multiple-choice problem or an extractive question answering problem.

When analyzing the explanations obtained with *LIME* for the same test example, it has been seen that *LIME* is able to detect the words of the label being explained, but it is curious that an *a priori* important word like "chats" it is not weighted by *LIME*, nor positive nor negative, with all experiments and modifications done. This is

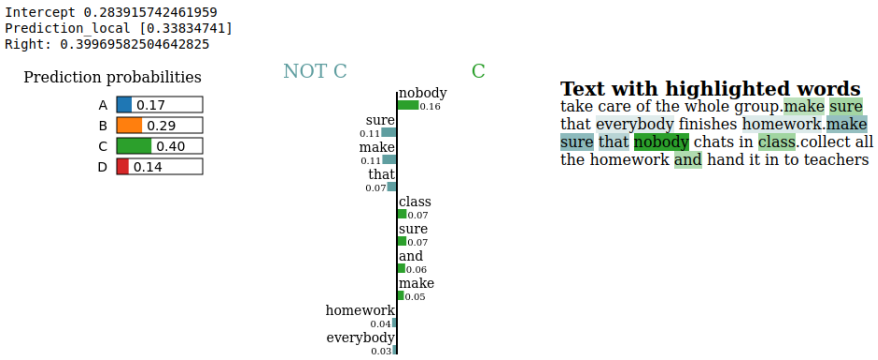



Figure 4.5: *LIME* result for the second experiment.

curious, because the other words of the options are common and/or related to the same domain, the school, meanwhile "chats" is a verb of a different domain -and more related to "discipline", the most important word of the question, but "nobody", which is also a "different" word and related in someway to "discipline" it is detected and it's detected as the most important word for the model, which makes sense.

In the third experiment the article was perturbed to see if *LIME* was able to find the most important words within it. The results showed that *LIME* knows how to deal

with large texts (inputs), but as the model's input size is not too large (due to hardware limitations in the training phase), the most important words of the text can not be captured by *LIME*.

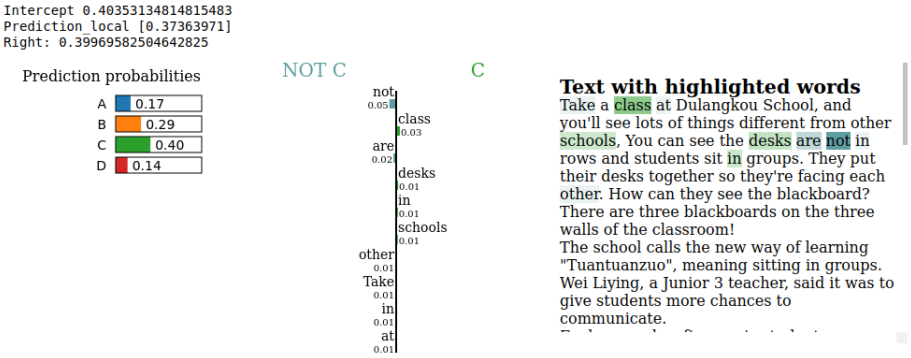



Figure 4.6: *LIME* result for the third experiment.

Even when this is obviously a problem, the experiment is not a complete failure, because it has been seen one of the most possible causes for the model to fail, this is, the input size is not as large as it should. For this kind of problems in which the input are large texts (such articles), the input size plays an important role in the results.

Nevertheless the model is still able to choose the correct option sometimes. This could be due to the knowledge the model has about the language. In these cases, where the model can't see the sentences the evidence is into, the model knows which one is the correct answer by understanding what the question requires and the meaning of the important words of the question and linking them to the most similar answer.

In the final experiment, both the question and the options are going to be analyzed. The results showed that the model focus more on the options than in the question. Even when explaining a specific label, *LIME* returns a greater weight for words in other options than in the question itself. This could mean that, if that words of other options

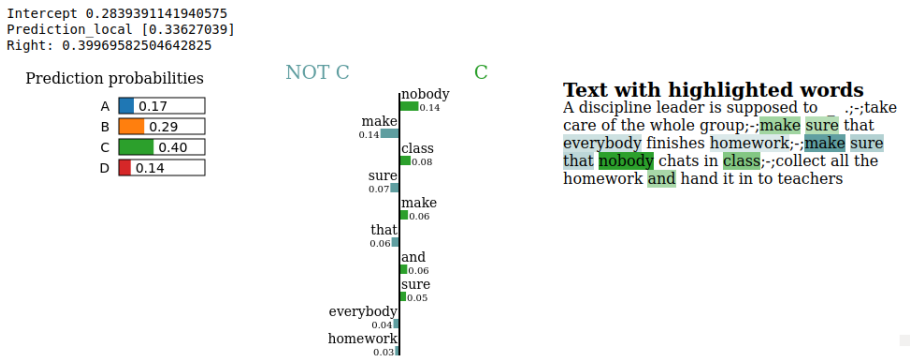



Figure 4.7: *LIME* result for the fourth experiment.

are changed, the model could choose them as the correct answer.

## 4.2. Model-specific experiments

### 4.2.1. Attention Heatmap

As said before, to visualize the *Attention* values as a heatmap is a common method (first introduced in the original *Attention* paper, (Bahdanau, Cho, y Bengio, 2014). In this case a simple heatmap has been plotted to see how *tokens* at the question attend to *tokens* at each of the options. As the *attention* heatmap takes the *attention* values of each of the heads, it has been decided to make the sum of the values of each head to get a unique value of each *token* relationship. This way, the heatmaps of figures 4.8 and 4.9 shows the sum of the *attention* values of each head of *tokens* in x-axis attending to *tokens* in y-axis.

As can be seen in Fig 4.8, the *discipline token* that is the most important one (according to other methods and having into account that is the *token* that differs with other types of leaders in the text) has not a special value or relationships with other important *tokens* in the sentence, such as *nobody chats in class*.

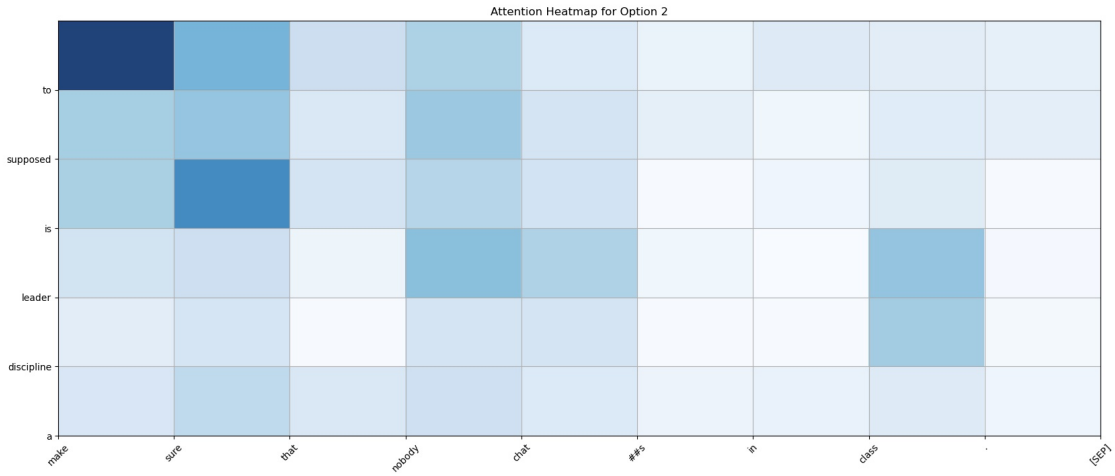
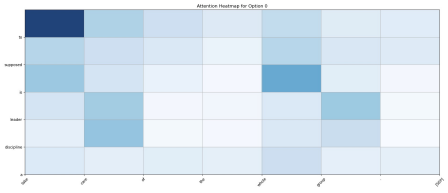
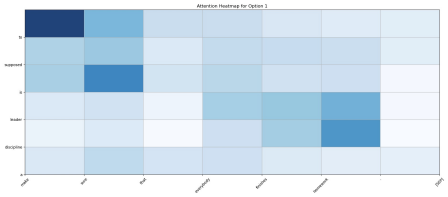



Figure 4.8: *Attention* heatmap for correct answer.

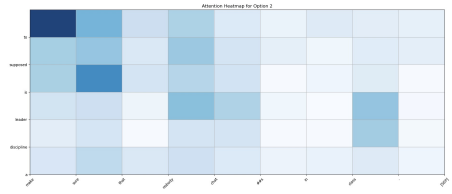
However, when looking at the heatmaps of other options, Fig 4.9, it can be seen that the *discipline token* has higher values when attending to *homework*, something that can have learn during the training, as someone with discipline is someone that always does the homework. Also when looking at the whole picture (heatmap of each option), it can be seen that there is no a special pattern or something that helps to understand the model decision.



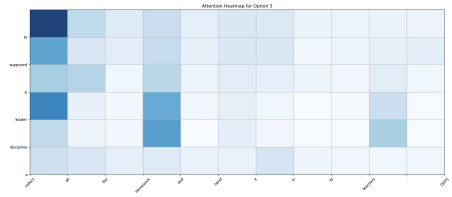
(a) *Attention* heatmap option A.



(b) *Attention* heatmap option B.



(c) *Attention* heatmap option C.



(d) *Attention* heatmap option D.

Figure 4.9: *Attention* Heatmaps.

### 4.2.2. BertViz

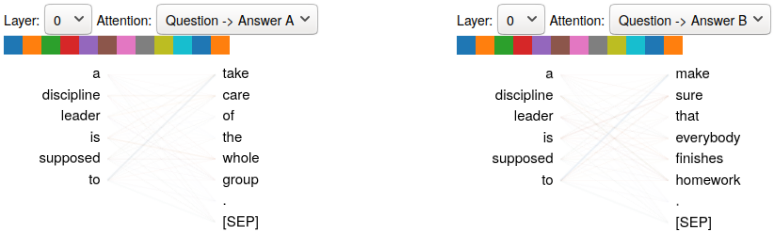
Due to the problems of *BertViz* to deal with large inputs, only has been tested the relations regarding the question and a single answer. This is, however, enough to test the tool, because it works in the same way no matter the input and, as it does not have to process anything, it does not matter the input used to test the tool and see how it works and how much information provides.

The first and most common way to study the *Attention* layers is to use the *Head View*, in which *BertViz* shows the *Attention* weights of a single layer. Again, the example used to test the tool has been: *A discipline leader is supposed to _ .*

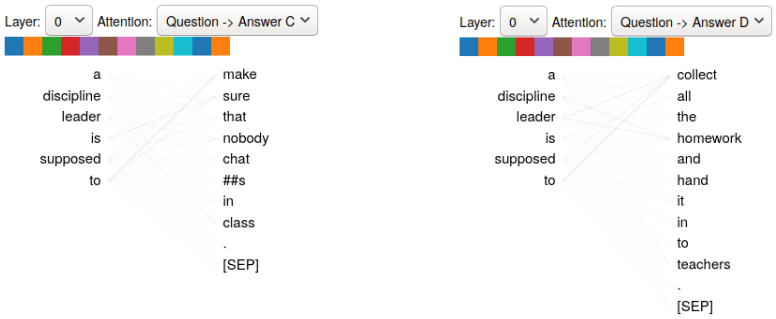
When looking at how *tokens* in Question attend to *tokens* in each Answer, it can be seen that there is not anything that clarifies why the model chooses an option among the other. Figure 4.10 shows layer 0 for each option. It can be seen that, for instance, *discipline* attends more to *tokens* such as *class* or *homework*, what makes sense.

For instance, it can be seen in Figure 4.11 that *discipline* attends to *make sure that nobody*, which is really related to *discipline*. However, there are different patterns across the layers and heads, and each head in each layer attend to *tokens* by following a specific pattern.

As there are usually too many heads (model used has 12 heads per each one of the 12 layers), it costs a lot to look at each one separately. In order to take a global look and to discover patterns and useless heads, the *BertViz Model View* is used. In Figure 4.12 it can be seen that the *Attention* layers follow patterns, even when the option change.




(a) Question attending to Answer A. (b) Question attending to Answer B.




(c) Question attending to Answer C. (d) Question attending to Answer D.

Figure 4.10: *BertViz* Head View of Question to Answers.

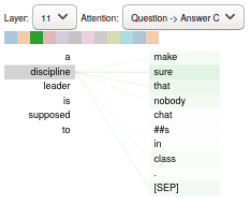
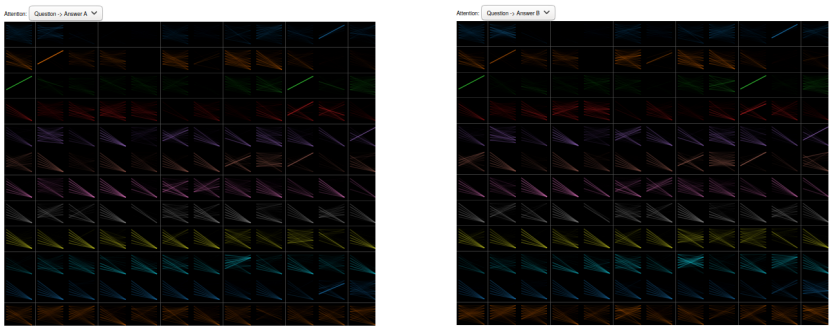



Figure 4.11: *BertViz* layer 11, head 2.



(a) Question attending to Answer A. (b) Question attending to Answer B.

Figure 4.12: *BertViz* Model View of Question to Answers A and B.

Once that a pattern have been detected (or an outlier) it's useful to look at that specific head. The tool let the user to do this by clicking in the corresponding head, Figure 4.13.

This way the user can see what *tokens* are being attended at in the patterns, e.g. nouns, prepositions, etc.

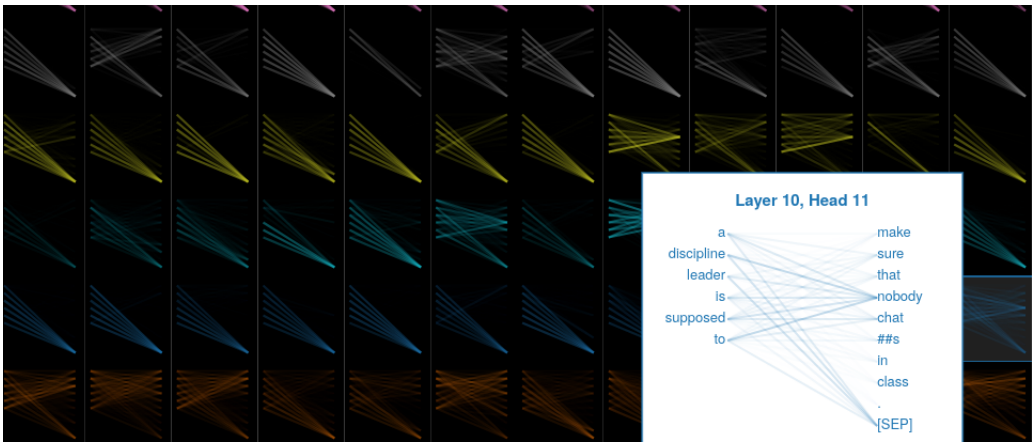



Figure 4.13: To look at a single head in the Model View of *BertViz*.

The model view is useful to have a global look of the model, to see its behaviour and to see what heads are useful/useless. For instance, it can be seen that a lot of heads follow the same pattern, all *tokens* attend to the *[SEP] token*, what is considered as a *null head*. These heads, therefore, could be prune to save space and time. This could be due to the training of the model, meaning that the model could be underfitted and, therefore, should be trained with more data or during more time, which in this case is true, because due to hardware and temporal limitations the model has been training for just 1 day, which is not enough for a *LM*.

Therefore, to look at the *Attention* does not really provide an explanation of why the model choose one option, because it follows patterns and does not change over the different options.

Although *BertViz* does not help to explain a prediction of the model, it is a good tool for visualizing the *Attention* weights, what can be useful as said before to, for instance, prune useless heads. But it's not a good option for explain a prediction of the model.

### 4.2.3. Intrinsic GPT-3

In this case, three passages were selected from the test set to feed *GPT-3* and force it to generate the text as wished, this is, an explanation of the answer. These passages were complex to answer, because in some of them the answer was not in the text itself, but related to. For instance, in one of the passages the question asks about the location of the story, given as options: England; America; Japan; Australia. But in the text it talks about New York, so a bit of commonsense is needed in order to relate New York with America.

The examples feed to the model included the context, the question, the options, the correct answer and a simple explanation of why it is the correct answer. The appendix A includes all the examples feed to the model.

**Example 1** The first try was using the Example 1, **??**. As the answer was in the text it was not surprising that *GPT-3* predicted it correctly. Regarding the explanation, Figure 4.14, it is a good explanation but, again, as the answer is inside the text there was not too much to explain but say that the article says so.

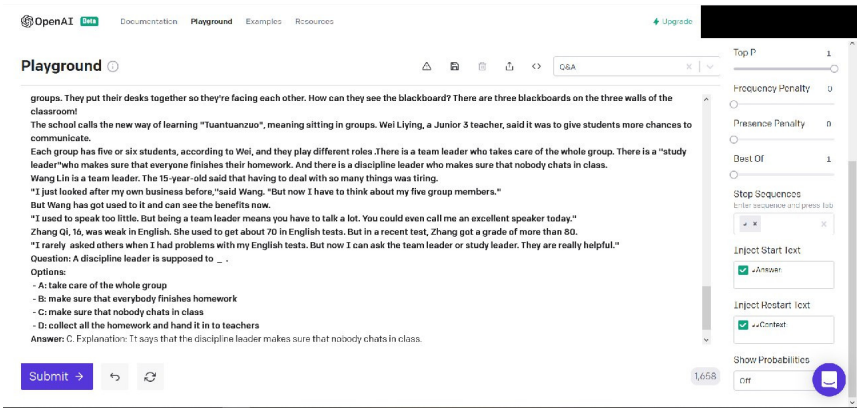



Figure 4.14: Explanation of GPT3 for the Example 1.

**Example 2**

Surprisingly, the result of *GPT-3* included the correct answer and an accurately explanation, 4.15.

But it has been detected that it is not always that good, and therefore it is not clear if it's indeed able to explain itself. With a simple change of options, interchanging the options A and B, the result fails in the prediction, the explanation is the same though, 4.16. And if an extra "answer" token is added to the end, the result of the prediction is correct but it fails in the explanation, 4.17.

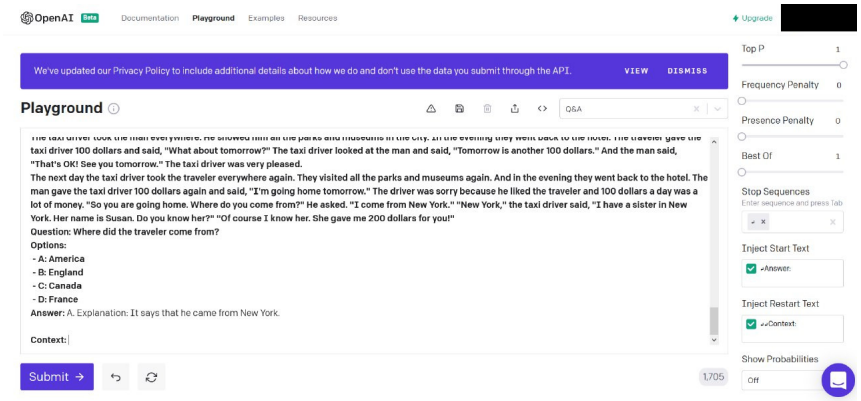



Figure 4.15: Explanation of GPT3.

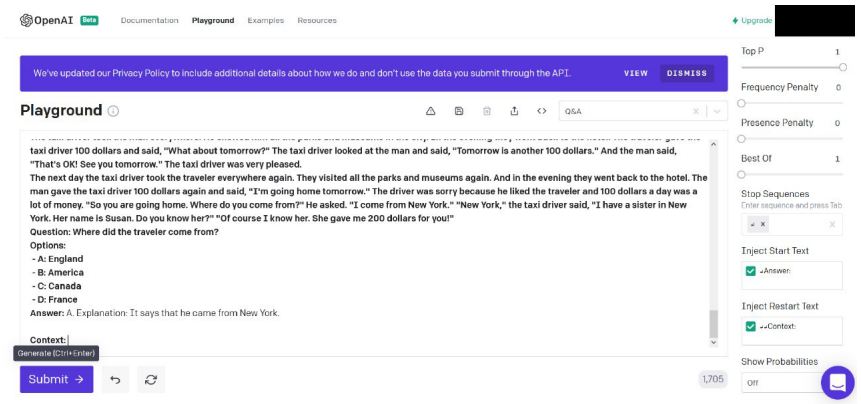



Figure 4.16: Wrong prediction of GPT3.

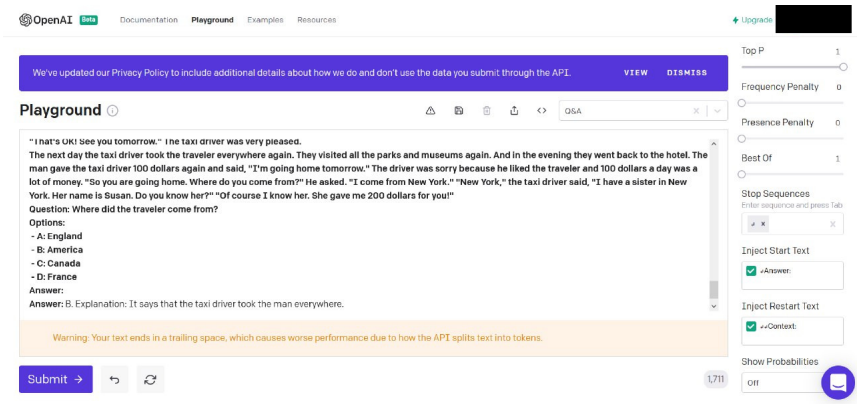



Figure 4.17: Wrong explanation of GPT3.

# Chapter 5

# Conclusions and Future Work

## 5.1. Conclusions

To explain *black box* models such as deep learning models it is a really complex task that, despite of the efforts of the researchers, is still a challenge far away from being completely solved.

Among the last years researchers have developed some interesting approaches that are really helpful to understand the model behavior and to know a bit more about why the model has given an output instead of another one. These approaches are based on some different techniques that help the user to understand what is going on inside the model when giving an output, and to make it easier for the user are often presented with visualization tools and techniques that make really simple to understand the explanations.

**LIME** After several experiments, it can be seen that *LIME* is able to detect the most important words (or spans) the model focus on when making its prediction. But it's still not clear why the model makes a prediction, because *LIME* explains the most important words for each class, but does not explain why the model choose a class over the other ones.

This way, *LIME* can help to understand the words that are important for the prediction, an this can help to understand the best way to create questions or to give the model the correct options. But it have been also seen that to change this words or this options does not mean the model is going to change its prediction. For instance, in the experiments in which the most important word "discipline" was replaced with a synonym "orderliness", the model predicted the good label meanwhile with the word "discipline" that appeared in the original text the model did not, and in the experiment in which the options given to the model were perturbed trying to preserve all the important words in the original question (like "supposed") the model also failed in its prediction.

So it is not clear which one is the correct way to make a question, or why the model has chosen a class instead of another one. Besides this, when the model fails, *LIME* does not really help to know why the model has failed, because the explanations of each of the classes will give us the important words for that class, but will not tell why a class is chosen beyond the others.

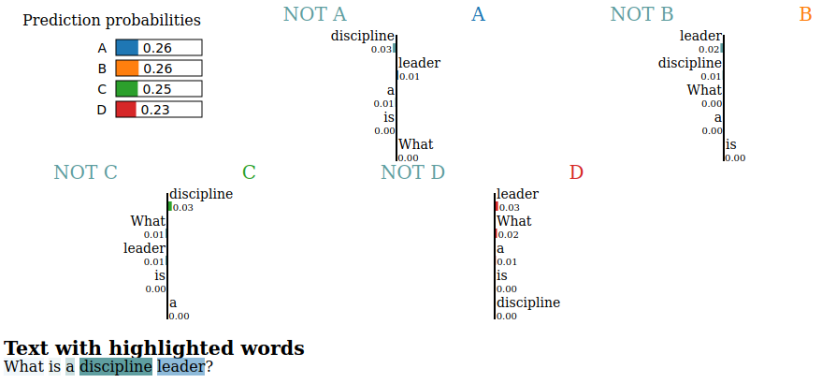



Figure 5.1: Explanation of all options given by *LIME*

**BertVIZ** *BertViz* is a tool to explore and visualize *attention* weights in every *transformer-based* model. It allows the user to visualize *attention* weights in an easy and user-friendly way, making possible to see and detect bias and patterns in the *attention* mechanism as well as to detect useless heads to be able to prune them and save space and time without loosing performance.

But, *BertViz* is just that, a visualization tool. It does not provide or perform any additional operation over the model and, therefore, it does not help more than other visualization tools to understand what is going on inside the model. Although, as said before, it helps to visualize *attention* weights to, for instance, detect patterns or see how words relate with others, this is not helpful when trying to understang the model behaviour and to know why the model has given an output and why not another one. But *BertViz* is not guilty for this, because as some research articles() has pointed out recently, to visualize the *attention* weights does not really provide an explanation of the local prediction, because the *attention* mechanism is just used to generate contextual embeddings of the input. Instead of that it would be more useful to explain the final layers of the model, that are the ones that are really solving the task, but this is very difficult, as they have as input the embeddings generated by the full model, making it really difficult to explain the layers behavior and to understand the explanations.

Therefore, other techniques such as *LIME* that create a surrogate explainable model are more useful than to visualize the *attention* weights in order to explain the model behavior.

**GPT-3** Despite of having some good explanations from *GPT-3* explaining itself, it is not clear if *GPT-3* is able to explain itself. When the explanation is in the article or it's easy to explain, it has not problem to do it. But when the question or the explanation are more difficult, not always give a valid explanation (sometimes nor even a good answer).

Although the explanations are not always valid, they are gramatically correct, taking sentences from the article. But sometimes they are not even related to the question, other times it gives the correct explanation, as in 4.15, where the option was *America* and the explanation is: *It says that he came from New York*, which is indeed in America but the lack of explanation relating New York to America could make it a bad explanation that is not enough for a user to understand the result, as seen in 2.2.1, the completeness was one of the *criteria* used by (Allam y Haggag, 2012) to select an answer as correct. Did the model relate New York with America to choose it as the result and the explanation? or did it just focus on that the question was asking for a place? This is not clear as the explanation was not consistent and not complete. So, this explanation, as seen in 2.2.1 and following the classification of (Pablo-Sánchez, 2020), this answer could be classified as incomplete. But this classification was thought for answers of *QA* systems, not for explanations. As seen in **??** there is not a common way to evaluate the explanations what can lead to a lack of quality in the results. For instance in this case, in which *GPT-3* is giving an explanation in natural language, it is easier to understand for the user, but the explanation is not consistent nor complete, so the answers classification could be used for this type of explanations.

## 5.2. FutureWork

# References

## Bibliografía

[Ahmed y Anto2016] Ahmed, Waheeb y Babu Anto. 2016. Answer extraction and passage retrieval for question answering systems. *International Journal of Advanced Research in Computer Engineering and Technology*, 5:2703–2706, 12.

[Allam y Haggag2012] Allam, Ali y Mohamed Haggag. 2012. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences*, 2:211–221, 09.

[Alvarez-Melis y Jaakkola2017] Alvarez-Melis, David y Tommi Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. En *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, páginas 412–421, Copenhagen, Denmark, Septiembre. Association for Computational Linguistics.

[Aubakirova y Bansal2016] Aubakirova, Malika y Mohit Bansal. 2016. Interpreting neural networks to improve politeness comprehension. *ArXiv*, abs/1610.02683.

[Bahdanau, Cho, y Bengio2014] Bahdanau, Dzmitry, Kyunghyun Cho, y Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. sep.

[Bengio et al.2001] Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, y Christian Jauvin. 2001. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Brown et al.2020] Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, y Dario Amodei. 2020. Language Models are Few-Shot Learners. may.

[Carton, Mei, y Resnick2018] Carton, Samuel, Qiaozhu Mei, y Paul Resnick. 2018. Ex-

tractive adversarial networks: High-recall explanations for identifying personal attacks in social media posts.

[Clark et al.2020] Clark, Kevin, Minh-Thang Luong, Quoc V. Le, y Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. mar.

[Danilevsky et al.2020] Danilevsky, Marina, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, y Prithviraj Sen. 2020. A survey of the state of explainable ai for natural language processing.

[Devlin et al.2018] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, y Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. oct.

[Ghaeini, Fern, y Tadepalli2018] Ghaeini, Reza, Xiaoli Fern, y Prasad Tadepalli. 2018. Interpreting recurrent and attention-based neural models: a case study on natural language inference. En *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, páginas 4952–4957, Brussels, Belgium, Octubre-Noviembre. Association for Computational Linguistics.

[González2003] González, José. 2003. La búsqueda de respuestas: Estado actual y perspectivas de futuro. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, ISSN 1137-3601, null 8, No. 22, 2004, pags. 37-56*, 8, 01.

[Gupta y Schütze2018] Gupta, Pankaj y Hinrich Schütze. 2018. LISA: Explaining recurrent neural network judgments via layer-wIse semantic accumulation and example to pattern transformation. En *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, páginas 154–164, Brussels, Belgium, Noviembre. Association for Computational Linguistics.

[Jain y Wallace2019] Jain, Sarthak y Byron C. Wallace. 2019. Attention is not Explanation. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, páginas 3543–3556, Minneapolis, Minnesota, Junio. Association for Computational Linguistics.

[Kneser y Ney1995] Kneser, Reinhard y Hermann Ney. 1995. Improved backing-off for m-gram language modeling. En *1995 International Conference on Acoustics, Speech, and Signal Processing*, volumen 1, páginas 181–184. IEEE.

[Lai et al.2017] Lai, Guokun, Qizhe Xie, Hanxiao Liu, Yiming Yang, y Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. En *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, páginas 785–794, Copenhagen, Denmark, Septiembre. Association for Computational Linguistics.

[Lecue2020] Lecue, Freddy. 2020. On the role of knowledge graphs in explainable ai. *Semantic Web*, 11(1):41–51.

[Lei, Barzilay, y Jaakkola2016] Lei, Tao, Regina Barzilay, y Tommi S. Jaakkola. 2016. Rationalizing neural predictions. *CoRR*, abs/1606.04155.

[Lertvittayakumjorn y Toni2019] Lertvittayakumjorn, Piyawat y Francesca Toni. 2019. Human-grounded evaluations of explanation methods for text classification. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, páginas 5195–5205, Hong Kong, China, Noviembre. Association for Computational Linguistics.

[Lewis et al.2019] Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, y Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.

[Li y Roth2002] Li, Xin y Dan Roth. 2002. Learning question classifiers. En *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING 02, páginas 1–7, USA. Association for Computational Linguistics.

[Ling et al.2017] Ling, Wang, Dani Yogatama, Chris Dyer, y Phil Blunsom. 2017. Program induction by rationale generation : Learning to solve and explain algebraic word problems.

[Luo et al.2018] Luo, Ling, Xiang Ao, Feiyang Pan, Jin Wang, Tong Zhao, Ningzi Yu, y Qing He. 2018. Beyond polarity: Interpretable financial sentiment analysis with hierarchical query-driven attention. En *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, páginas 4244–4250. International Joint Conferences on Artificial Intelligence Organization, 7.

[Mikolov et al.2013] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, y Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. En *Advances in neural information processing systems*, páginas 3111–3119.

[Moldovan et al.2001] Moldovan, Dan, A Harabagiu, Marius Paca, Rada Mihalcea, Richard Goodrum, Roxana Girju, y Vasile Rus. 2001. Lasso: A tool for surfing the answer net. 11.

[Moon et al.2019] Moon, Seungwhan, Pararth Shah, Anuj Kumar, y Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, páginas 845–854.

[Openai et al.] Openai, Alec Radford, Karthik Narasimhan Openai, Tim Salimans Openai, y Ilya Sutskever Openai. Improving Language Understanding by Generative Pre-Training. Informe técnico.

[Pablo-Sánchez2020] Pablo-Sánchez, César. 2020. Bootstrapping named entity resources for adaptive question answering systems. 04.

[PALMONARI y MINERVINI2020] PALMONARI, Matteo y Pasquale MINERVINI. 2020. Knowledge graph embeddings and explainable ai. *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges*, 47:49.

[Papernot y McDaniel2018] Papernot, Nicolas y Patrick D. McDaniel. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *CoRR*, abs/1803.04765.

[Pappas y Popescu-Belis2014] Pappas, Nikolaos y Andrei Popescu-Belis. 2014. Explaining the stars: Weighted multiple-instance learning for aspect-based sentiment analysis. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, páginas 455–466, Doha, Qatar, Octubre. Association for Computational Linguistics.

[Poerner, Schütze, y Roth2018] Poerner, Nina, Hinrich Schütze, y Benjamin Roth. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. En *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 340–350, Melbourne, Australia, Julio. Association for Computational Linguistics.

[Radev et al.2002] Radev, Dragomir, Weiguo Fan, Hong Qi, Harris Wu, y Amardeep Grewal. 2002. Probabilistic question answering on the web. En *Proceedings of the 11th International Conference on World Wide Web*, WWW 02, página 408419, New York, NY, USA. Association for Computing Machinery.

[Radford et al.2019] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, y Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. Informe técnico.

[Raffel et al.2019] Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, y Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. oct.

[Rajani et al.2019] Rajani, Nazneen Fatema, Bryan McCann, Caiming Xiong, y Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, páginas 4932–4942, Florence, Italy, Julio. Association for Computational Linguistics.

[Rajpurkar, Jia, y Liang2018] Rajpurkar, Pranav, Robin Jia, y Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

[Rajpurkar et al.2016] Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, y Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

[Ribeiro, Singh, y Guestrin2016] Ribeiro, Marco Tulio, Sameer Singh, y Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. En *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, páginas 1135–1144.

[Rogers et al.2020] Rogers, Anna, Olga Kovaleva, Matthew Downey, y Anna Rumshisky. 2020. Getting closer to ai complete question answering: A set of prerequisite real tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8722–8731, Apr.

[Rosset2020] Rosset, C. 2020. Turing-nlg: A 17-billion-parameter language model by microsoft.

[Serrano y Smith2019] Serrano, Sofia y Noah A. Smith. 2019. Is attention interpretable? En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, páginas 2931–2951, Florence, Italy, Julio. Association for Computational Linguistics.

[Shoeybi et al.2019] Shoeybi, Mohammad, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, y Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

[Sydorova, Poerner, y Roth2019] Sydorova, Alona, Nina Poerner, y Benjamin Roth. 2019. Interpretable question answering on knowledge bases and text. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, páginas 4943–4951, Florence, Italy, Julio. Association for Computational Linguistics.

[Taylor1953] Taylor, Wilson L. 1953. Cloze procedure: A new tool for measuring readability. *Journalism and Mass Communication Quarterly*, 30(4):415.

[Vaswani et al.2017] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, y Illia Polosukhin. 2017. Attention is all you need. jun.

[Vig2019] Vig, Jesse. 2019. A multiscale visualization of attention in the transformer model. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, páginas 37–42, Florence, Italy, Julio. Association for Computational Linguistics.

[Voorhees2000] Voorhees, Ellen. 2000. The trec-8 question answering track report. 11.

[Voskarides et al.2015] Voskarides, Nikos, Edgar Meij, Manos Tsagkias, Maarten de Rijke, y Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. En *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, páginas 564–574, Beijing, China, Julio. Association for Computational Linguistics.

[Wallace, Feng, y Boyd-Graber2018] Wallace, Eric, Shi Feng, y Jordan Boyd-Graber. 2018. Interpreting neural networks with nearest neighbors. En *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, páginas 136–144, Brussels, Belgium, Noviembre. Association for Computational Linguistics.

[Wu et al.2016] Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, y Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

[Xie et al.2017] Xie, Qizhe, Xuezhe Ma, Zihang Dai, y Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. En *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 950–962, Vancouver, Canada, Julio. Association for Computational Linguistics.

[Yamada et al.2020] Yamada, Ikuya, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, y Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention.

[Yan et al.2020] Yan, Yu, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, y Ming Zhou. 2020. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training. jan.

# Appendix A

# Examples feed to GPT-3

As said before, to force *GPT-3* to output an expected result, it has to be feed with some examples of the desired output to know what has to generate. In order to force it to generate explanations of the prediction, first a few examples of the test set has been formatted and added an explanation as desired. These are the full examples feed to *GPT-3*:

**Passage 1.1: Article of GPT Example 1**

It was the golden season. I could see the yellow leaves dancing in the cool wind. I felt lonely and life is uninteresting. But one day, the sound of a violin came into my ears. I was so surprised that I ran out to see where it was from. A young girl, standing in the wind, was lost in playing her violin.

I had never seen her before. The music was so wonderful that I forgot who I was. Leaves were still falling. Every day she played the violin in the same place and I was the only listener. It seemed that I no longer felt lonely and life became interesting. We didn't know each other, but I thought we were already good friends. One day, when I was listening, the sound suddenly stopped. The girl came over to me.

"You must like violin." she said.

"Yes. And you play very well. Why did you stop?" I asked.

Suddenly, a sad expression appeared on her face and I could feel something unusual.

"I came here to see my grandmother, but now I must leave. I once played very badly. It is your listening every day that has _ me." she said.

"In fact, it is your music that has given me those meaningful days." I answered. "Let us be friends."

The girl smiled and I smiled.

I never heard her play again in my life. Only thick leaves were left behind. But I will always remember the girl. She is like a dream; so short, so bright that it makes life beautiful.

There are many kinds of friends. Some are always with you, but don't understand you. Some say only a few words to you, but are close to you. I shall always think

of those golden days and the girl with the violin. She will always bring back the friendship between us. I know she will always be my best friend.

Q: "*What's the best title for the passage?*"
Options:

- A: A Musical Girl
- B: Wonderful Music
- C: A Special Friend in a Special Autumn
- D: How to Be Friends

Answer: C.
Explanation: It says that it was the golden season and autumn is known as the golden season and it speaks about friendship not just music.

**Passage 1.2: Article of GPT Example 2**

Is it important to have breakfast every day? A short time ago, a test was given in the United States. People of different ages, from 12 to 83, were asked to have a test. During the test, these people were given all kinds of breakfast, and sometimes they got no breakfast at all. Scientists wanted to see how well their bodies worked after eating different kinds of breakfast.
The results show that if a person eats a right breakfast, he or she will work better than if he or she has no breakfast. If a student has fruit, eggs, bread and milk before going to school, he or she will learn more quickly and listen more carefully in class. Some people think it will help you lose weight if you have no breakfast. But the result is opposite to what they think. This is because people become so hungry at noon that they eat too much for lunch. They will gain weight instead of losing it.

Q: "*According to the passage, what will happen to you if you don't have any breakfast?*"
Options:

- A: To be healthier.
- B: To work better.
- C: To gain weight.
- D: To fail the test.

Answer: C.
Explanation: Because it says that people become so hungry at noon that they eat too much for lunch.

**Passage 1.3: Article of GPT Example 3**

It's the second time for me to come to Beijing. There are many places of interest in Beijing, such as the Summer Palace, the Great Wall, etc. What's more, I think great changes have taken place in Beijing. People's living conditions have improved a lot. Their life is very happy. Almost everyone has a big smile on the face. People in Beijing are in high spirits and hard-working. Children can receive a good education. But in the past, some children didn't have enough money to go to school. They often worked for cruel bosses. The bosses didn't give them enough food. I feel sorry for them. Today people have already lived in tall building, worn beautiful clothes and so on. Life has changed greatly.

Q: "*How many times has the writer been to Beijing?*"
Options:

- A: Once.
- B: Twice.
- C: Three time.
- D: Four times.

Answer: B.
Explanation: Because it says that is the second time that he goes to Beijing.

**Passage 1.4: Article of GPT Example 4**

Several years ago,a television reporter was talking to three of the most important people. One was a very rich banker,another owned one of the largest companies in the world,and the third owned many buildings in the center of New York.
The reporter was talking to them about being important. "How do we know if someone is really important?" the reporter asked the banker.
The banker thought for a few moments and then said, "I think anybody who is invited to the White House to meet the President of the United States is really important."
The reporter then turned to the owner of the very large company. "Do you agree with that?" she asked.
The man shook his head, "No. I think the President invites a lot of people to the White House. You'd only be important if while you were visiting the President, there was a telephone call from the president of another country,and the President of the US said he was too busy to answer it."
The reporter turned to the third man. "Do you think so?"
"No, I don't," he said. "I don't think that makes the visitor important. That makes the President important. "

"Then what would make the visitor important?" the reporter and the other two men asked.
"Oh, I think if the visitor to the White House was talking to the President and the phone rang, and the President picked up the receiver, listened and then said, Ít's for you."'

Q: "*This story happened in _ .*"
Options:

- A: England.
- B: America.
- C: Japan.
- D: Australia.

Answer: B.
Explanation: It talks about New York, the United states and the White House that are in America.

**Passage 1.5: Article of GPT Example 5**

Dear John, Thank you very much for your letter. I am glad that you enjoyed your holiday with me. We enjoyed having you and your sister here. We hope that you will both be able to come again next year. Perhaps you'll be able to stay longer next time you come. A week is not really long enough, is it? If your school has a five-week holiday next year, perhaps you'll be able to stay with us for two or three weeks.
We have been back at school three weeks now. It feels like three months! I expect that you are both working very hard now that you are in Grade One. I shall have to work hard next year when I am in Grade One. Tom and Ann won't be in Grade One until 2011. They went for a picnic yesterday but I didn't go with them because I cut my foot and I couldn't walk very well. They went to an island and enjoyed themselves. Do you still remember the island? That's where all five of us spent the last day of our holiday. Tom, Ann and I send our best wishes to Betty and you. We hope to see you soon.
Yours sincerely,
Michael

Q: "*From the words of "It feels like three months!" we know that _ .*"
Options:

- A: Michael's teacher is very strict with the students.
- B: Michael is pleased with his school report.

- C: Michael has no interest in learning.
- D: Michael works very hard at his studies.

Answer: C.
Explanation: When something feels to go slower is because we don't like it. If we do like it seems to go faster.

## Todo list

*14*, ALBERT
*14*, LUKE
*19*, Introduction to QA Datasets
*23*, SWAG
*23*, ARC
*31*, Survey of XAI in QA