Detecting Salient Antimetaboles in English Texts Using Deep and Transfer Learning

Master Thesis

of

Guillaume Berthomet

109339

University of Passau

Chair of Data Science

National Institute of Applied Sciences of Lyon

Computer Science Division

Reviewers: Dr. Jelena Mitrović, Dr. Elöd Egyed-Zsigmond

Advisors: Dr. Diana Nurbakova, Ramona Kühn

January 19, 2023

Abstract

Computer Linguists have long looked at rhetorical devices and how best they could be computed by - and used for - Natural Language Processing (NLP) techniques. However, one such figure was barely studied until the last decade: the chiasmus. This Master Thesis will build upon the opening works of the last fifteen years while focusing on the detection of antimetaboles, a subcategory of chiasmi. Its aim is to study the effectiveness and accuracy of transfer learning regarding this task, using pre-trained transformers neural networks and to compare it to the results obtained using classical machine learning methods from the past research works. Specifically, we will show that transformers can offer very good results regarding this task but that those results are actually misleading, as they tend to look for literary properties that are not intrinsic to chiasmi. Finally, we will also share an annotated corpus of several hundred chiasmi, improving upon the previous corpora by an order of magnitude and opening up more possibilities for future researches.

Keywords - Computer Linguistics, Natural Language Processing, Chiasmus Detection, Deep Learning, Transformers

Acknowledgements

This Master Thesis has been a lot of work throughout the last year, and many people helped me make it possible. I would thus like to thank everyone who has played a role in the development and completion of this project.

First of all, I would like to express my gratitude towards the Département Informatique (Computer Science Department) of the INSA Lyon and the Fakultat für Informatik und Mathematik of the Universität Passau: their collaboration made my double degree possible in the first place, their teachings gave me a very solid base for my researches and their help and efforts eased the arduous task that is navigating a double degree and Master Thesis.

Moreover, I want to thank Dr. Jelena Mitrović and Ramona Kühn, for offering me the possibility to work on the fascinating subject of NLP and for all the help and advice they provided over the course of my Master Thesis, be it for technical questions about NLP or tips on how to properly conduce a thesis. I am also deeply grateful to Dr. Diana Nurbakova for the constant help and support she provided, always present when I was unsure on how best to approach problems and situations.

I would also like to acknowledge Yohan Meyer, a student with whom I shared the introductory part of my Master Thesis under the supervision of Jelena, Ramona and Diana. We worked together for the first several months of our theses, and his insight and contribution were always particularly helpful and to the point. His own thesis is referenced as Meyer (2023).

I am grateful to Pr. Randy A. Harris as well for the time and interest he gave Yohan and me despite the timezone difference with Canada. He gave us interesting and thought-provoking advice on classical linguistics and both his extensive work on rhetorical devices. The suggestions he made when conversing with us allowed us to see the subject with more depth and breadth than if we had approached it from a pure Computational Linguistics point of view.

I wish to extend my gratitude to Nikolaj Polle and Dr. Diana Nurbakova too, for the time they offered when proofreading my thesis. This work is long, and their help was precious to me, to make sure I would deliver a thesis of the highest possible quality.

I would furthermore like to thank my family for the love and support they provided, and my partner Nikolaj in particular for his continuous support and the help he offered me in the day-to-day process of organising, writing and working on this thesis.

Finally, I want to show appreciation to all those reading this work for their precious time and attention.

Contents

1 Introduction

Natural Language Processing ("NLP") has progressed steeply over the last two decades and is now offering useful results in several fields, with even more promising ones currently being researched. Among all those fields, one in particular is the detection in texts of interesting *rhetorical figures* - also known as *figures of speech*, or *rhetorical devices*. The interest of this rhetorical figures detection task is twofold: first, such rhetorical figures can be used by further NLP processes such as (but not limited to) sentiment analysis (e.g. Yadav and Vishwakarma (2020)) or essay grading (e.g. Ramesh and Sanampudi (2022)); second, it offers in and of itself further understanding on how to process natural language, thanks to its reliance on all three levels of language: lexical, grammatical and semantic.

Being an integral piece of human communication, rhetorical figures have interested computational linguists for as long as the field has existed. An example would be the book *Processing Metonymy and Metaphor* (Fass, Lesgold, & Patel, 1997), which was already trying to study and explain how those two figures could be best understood, classified and computationally processed twenty-five years ago. But the focus of such studies often fell on the most common devices of speech while some were barely studied, if at all, until much more recently.

Such is the case of the *chiasmus*, an excessively rare rhetorical figure in the English language[1], which may explain the lack of research around them. Chiasmi are a particularly interesting figure to study, but also a high challenge for any NLP system as they exist at the border between *tropes* and *schemes*. *Tropes* are the rhetorical devices which focus on the meaning of words (e.g. metaphors or oxymorons), while *schemes* are the devices which focus on patterns within the language (e.g. parallelism or anaphora).

This in-between makes chiasmi particularly hard to use but also particularly potent figures when properly employed, as they are able to efficiently and aesthetically convey either similarity or opposition between two clauses. Being able to detect them with a high degree of confidence could thus improve various NLP tasks focusing on literary works, especially when combined with other quality markers such as grammatical structure or vocabulary analysis.

As we will explore more deeply in Chapter 3, extensive work on the detection of chiasmi only began a decade ago with the pioneering works of Dubremetz and Nivre during the conduct of Marie Dubremetz's Ph.D (2014 - 2017). Since then however, the subject of

[1] Dubremetz and Nivre (2015) used the book *River War* by Winston Churchill to exemplify the rarity of chiasmi: only one instance of a purposeful, rhetorically salient chiasmus can be found in the more than one hundred thousand words of this work written by an excellent rhetorician.

the chiasmus was only barely explored by the NLP community: only one other article was published, Schneider, Barz, Brandes, Marshall, and Denzler (2021), which became the state of the art by detecting antimetaboles with the same rate as Dubremetz and Nivre (2017) while also detecting some harder to catch semantic chiasmi.

All of those works were led using tools from classical statistical theory and machine learning, and we thus decided for this thesis to explore the possible uses of deep learning methods to solve the problem of salient chiasmi detection. In particular, this work aims to study **how salient antimetabole could be detected using *transformers*** (refer to Wolf et al. (2020a) for a survey), a subcategory of deep neural networks, as opposed to the deterministic algorithms or classical machine learning methods which were used up until now.

We will hence show that transformers can be highly effective to detect such rhetorical figures, but that this effectiveness is often misleading - partly at least - and actually a result of a bias toward features not directly related to antimetaboles. We will then discuss how this bias could be mitigated in order for transformers to look at more inherent qualities of antimetaboles instead of focusing on extrinsic properties that can, and do, mislead their results.

Another important contribution of this thesis is the large amount of chiasmi data we gathered. **A large dataset** composed of hundreds of salient chiasmi, thousands of random criss-cross patterns (i.e. non-salient chiasmi) and thousands of random sentences from a large array of sources have been compiled and is now freely available for anyone who wish to pursue researches on chiasmi and antimetaboles.

1.1 Organisation of the Thesis

This thesis will be divided into five main parts: first, we will explore the technical background necessary to properly understand the terms and techniques used in this thesis. Then, we will explore the academic background of the task at hand, beginning with a general overview of the field of natural language processing before focusing on the specific task of chiasmus detection. Subsequently, we will focus on the methodology we followed throughout the thesis to obtain our results before presenting those results in a subsequent chapter. Finally, we will discuss those results before concluding with a general note on our works and findings, and propose some ideas for future works on this subject.

2 Technical Background

In this chapter, we will define and explain the most important points on which this thesis relies. More precisely, the first section will focus on linguistics, with the aim to formally define the figures we are looking into, while the second section will focus on computer science, giving the necessary technical background to understand the techniques of machine learning discussed in the following chapters.

2.1 Defining Chiasmi and Antimetaboles

Etymologically, the name "*chiasmus*" finds its origin from the Greek $\chi\iota\acute{\alpha}\zeta\omega$ - "to shape like the letter χ" - because the criss-cross pattern of its elements was linked with the cross-like shape of this antic letter. However, an etymological explanation of the word is not enough to delve deeper into a study of chiasmi and how computer could detect them. We can thus define a chiasmus as follow:

Definition 1. ***Chiasmus****: The inverse repetition of any two pairs of linguistic elements in a larger coherent body of text.*

Those inverse repetition are also sometimes described as a "criss-cross pattern" or a "chiastic pattern" in the text and can be visualised with Figure 1. Our definition of a chiasmus is purposefully vague, as previous definitions have up until now been varied, overlapping and sometimes even contradicting. For example, Dubremetz and Nivre (2017) define a chiasmus as "[a repeating] pair of identical words in reverse order" while Schneider et al. (2021) define a chiasmus as "an inversion of semantically or syntactically related words, phrases, or sentences". Therefore, we decided to take inspiration from Harris, Di Marco, Ruan, and O'Reilly (2018) for our work and aimed at a general and all-purpose definition for chiasmi, which could then be sub-divided into different types of chiasmi.

In particular, one such subtype of the chiasmus is the *antimetabole* on which this thesis will focus from now on. Antimetaboles can be defined as follow:

Definition 2. ***Antimetabole:*** *A "lexical chiasmus" (Harris et al., 2018), that is an inverse repetition of lexical elements named "lemmata".*

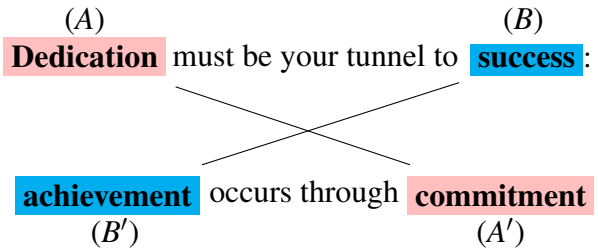

Figure 1: A visualisation of the criss-cross pattern in a (semantic) chiasmus

Definition 3. ***Lemma (plural lemmata or lemmas):*** *In linguistics, the canonical form of a given set of related words.*[2]

Antimetaboles have a wide range examples, but some of them are particularly known to the English speaking public such as Examples 1 and 2. In some cases, they even became prototypical of what it means to be an antimetabole in the general culture for their simplicity and efficiency - although it does not mean that antimetaboles are entirely restricted to their most prototypical examples and subtler ones definitely exist like Example 3.

- ***(1)*** ***One*** *for* ***all*** *,* ***all*** *for* ***one*** *.*[3]
- ***(2)*** ***Live*** *not to* ***eat*** *, but* ***eat*** *to* ***live*** *.*[4]
- ***(3)*** *The first half of life consists of the* ***capacity*** *to enjoy without the* ***chance*** *; the last half consists of the* ***chance*** *without the* ***capacity*** *.*

An additional, but very important point, is the question of what exactly is a *rhetorical figure* (Harris et al., 2018). Most would agree that Examples 1 to 3 are good examples of antimetaboles, while Example 4 below does not appear to contain any noteworthy chiastic repetition. However, if we go by our strict definition of antimetabole, Example 4 is indeed an antimetabole because of the inverse repetition on the lemmata "*I*" and "*plant*".

- ***(4)*** ***I*** *would like having* ***plants****, especially* ***plants*** *that fit well with the furniture* ***I*** *bought.*

[2] An example of a lemma would be *hunt*, which is the lemma associated with several words: "to hunt", "hunted", "(a) hunt", "hunts" and more.
[3] Alexandre Dumas, originally in French: "Un pour tous, tous pour un".
[4] Socrates (sometimes also attributed to Benjamin Franklin).

From now on, we will then refer to phrases like Examples 1 to 3 not simply as antimetaboles but as (rhetorically) salient antimetaboles, while phrases like Example 4 will be referred to as non-salient antimetabole. The reasoning behind this is that, while they all technically are antimetabole, only some of them have literary interest and should be detected. This work will therefore mostly focus on the detection of **salient** antimetaboles and their separation from non-salient ones.

2.2 A quick exploration of Machine Learning

Definition 4. ***Machine Learning (ML)***[5]*: "The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data."*

The link between NLP and ML runs deep, as the latter is nowadays one of the former's most used tool for a plethora of various tasks ranging from text preprocessing (Qi, Zhang, Zhang, Bolton, & Manning, 2020) to content summarization (Nallapati, Zhou, dos Santos, Gulcere, & Xiang, 2016) or quality evaluation (Wachsmuth, Al-Khatib, & Stein, 2016). ML can moreover be divided into several subcategories, with the main distinction drawn in this thesis being "Classical Machine Learning" (Classical ML) against "Artificial Neural Networks" (ANN) and - in particular - "Deep Learning" (DL). This specific distinction between Classical ML and ANN is interesting to us, as all works until now have used Classical ML to detect chiasmi (Dubremetz & Nivre, 2017; Schneider et al., 2021) whereas this work aims to study the effectiveness of Deep Learning models for the selfsame task.

2.2.1 Classical Machine Learning

Definition 5. ***Classical Machine Learning:*** *The set of statistical tools aiming to learn conclusions about a given dataset by using specific mathematical features, like the distance between data-points or the direction of vectors.*

[5]From the Oxford Language dictionary.

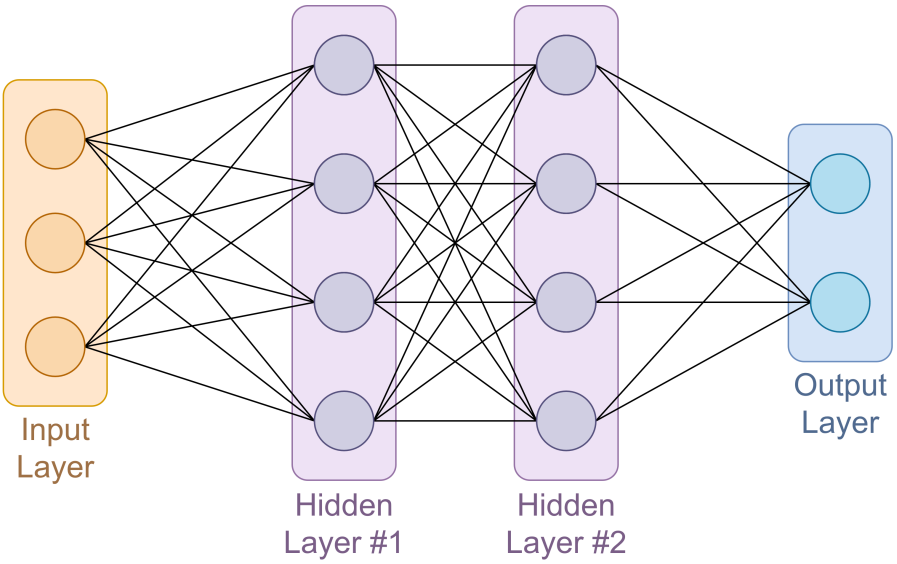

Figure 2: A visualisation of a simple fully connected ANN with one input layer, two hidden layers and one output layer.

Two particularly known examples - but far from the only ones - of Classical ML are:

- *Support Vector Machines* ("SVM", first presented in Cortes and Vapnik (1995) under the name of "Support Vector Networks") which aim to classify a collection of data points by translating them into a high-dimensional space before drawing a line maximising the distance between points in that space. This line can then be used to classify new data points either in "Set A" or "Set B".
- *Logistic Regression* (a mathematical method finding its roots in the 19th century with the works of P. F. Verhulst, between 1838 and 1847) which aim to determine the behavior of a set of points by trying to find an underlying mathematical function their distribution follows, and then use that function to try and predict the position of future points or classify new ones.

The common characteristic of Classical ML methods is that they study the available data as a whole to try and extract interesting mathematical features from it, with such underlying features being then used for prediction of future data points or classification of new ones.

2.2.2 Artificial Neural Networks

Definition 6. ***Artificial Neural Networks ("ANN"):*** *A network of interconnected nodes (see Figure 2) which aims to emulate a simplified version of animal brains to process inputs: the nodes are therefore called neurons, and the whole network a neural network.*

The idea of ANN originated with the perceptron, a modelisation of the visual processing within the human brain (Rosenblatt, 1958). ANN process their input features by forwarding them to a layer of "neurons", with each neuron being a combination of a linear function processing its inputs and a non-linear function transforming the linear function's output into a non-linear one (Figure 3 illustrates the working of an artificial neuron). This output is then forwarded to another layer of neuron, and this feed-forward chain can be repeated as many times as needed.

Once the information has reached the last layer of neurons, its output(s) are then used as the result of the computation. Different tasks can be emulated with different architectures of output layers: identification tasks only need one neuron, with its activation or not signifying true or false; classification tasks use as many neurons as there is possible classes, one for each; generation use as many neurons as is necessary to encode the generated result; etc...

This architecture allows ANN to emulate any mathematical functions, including highly non-linear ones thanks to the non-linear functions within neurons[6]. As all data processing tasks can be summed up to equivalent mathematical functions (although those functions are most of the time incredibly complex), any of them can theoretically be solved by a properly crafted ANN.

ANN are trained in a very different way than Classical ML tools: rather than looking at datasets as a whole, they only ever look at one (or a few) data point(s) at a time. A simplified but interesting way to understand this is as follow: during the training process, they compute for each new data point "how far" from the correct result their own results are before adjusting their neurons' inner parameters accordingly, i.e. so that their results would be closer to the correct one if they passed on the same data again.

In practice, ANN use a loss function when training: this function gives a loss value that quantifies the distance between the correct output and the predicted output. A method called backpropagation (D. Rumelhart, Hinton, & Williams, 1986)[7] (abbreviated from "backward propagation") is then used to adjust the network's parameters accordingly: a gradient of the loss function with respect to each of the parameters of the network is computed, beginning with the parameters of the last layer and using those results to propagate the gradient computation backward (hence its name). Finally, this gradient is used to adjust the parameters accordingly, edging them closer to the expected output. Computing the gradients with respect to a loss function instead of computing them directly with respect to the output makes the process much easier because of the reduced

[6] If neurons had a purely linear behavior, ANN could only emulate linear functions since any linear combination of linear functions is a linear function itself.

[7] We cite here D. Rumelhart et al. (1986) as they were the first to coin the term "backpropagation", although the idea itself was first studied in the sixties.

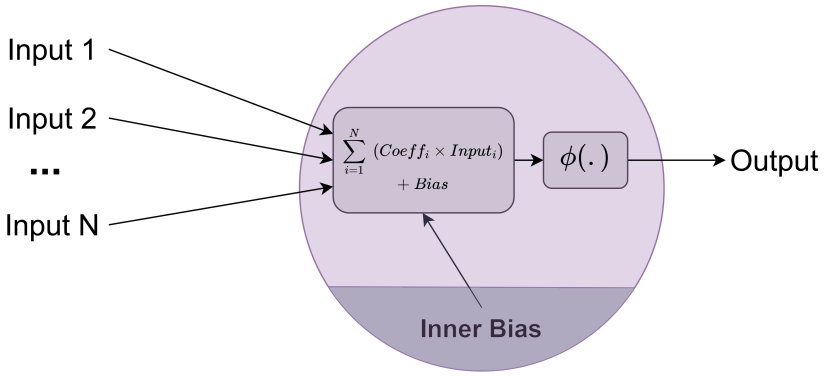

Figure 3: The inner working of an artificial neuron in an ANN. *Bias* and $Coeff_i$ are inner parameters, $\phi(.)$ is a non-linear function.

amount of dimension. The discovery and generalized usage of backpropagation is one of the key factor allowing neural networks to work as they do today.

2.2.3 Deep Neural Networks

Definition 7. ***Deep Neural Network ("DNN")****: An ANN with a high number[8] of neuron layers between the input and the output.*

Definition 8. ***Deep Learning ("DL")***[9]*: The training and use of Deep Neural Networks to execute tasks and solve problems.*

DNN were first discussed in Rosenblatt (1958), and the first theoretical algorithm for them was proposed by Lapa and Ivakhnenko (1967). They aim to solve problems which are not easy to mathematically define. Specifically, their high number of layers (called hidden layers, as they are not directly interacted with by external users) allows more control over the architecture of the network and thus help shaping it to better solve specific problems. Particularly well-known examples of Deep Learning methods are *Convolutional Neural Networks* ("CNN"; first named in LeCun, Bottou, Bengio, and Haffner (1998) although the concept was described as early as Fukushima and Miyake (1982)) and *Recurrent Neural Networks* ("RNN"; D. E. Rumelhart, Hinton, and Williams, 1986).

[8] An ANN is often considered "deep" when it has at least ten hidden layers.
[9] The name *Deep Learning* was first proposed by Dechter (1986)

In a CNN, some of the hidden layers are not made of neurons but instead perform a convolution operation of their inputs. This makes the network's pattern recognition abilities invariant to translation and scaling, which in turn helps in tasks such as the detection of features whose position and size are not fixed (e.g. objects within a larger image).
In a RNN, the output of "higher" layers when processing one data point is then used as input for "earlier" layers when processing the next data point which creates a form of "memory of the past" and allows them to better process strings of data points with a temporal relation between them.

2.2.4 Transformers

Transformers are a very recent addition to the family of Deep Learning algorithms, as they were proposed by Vaswani et al. (2017) only five years ago. However, in this short time frame, they have taken over more and more tasks that were previously run by other kind of Neural Networks, in particular in NLP in which they have become the *de facto* algorithms to use for most tasks. Transformers now enhance Google searches (Nayak, 2019) or allow machine to converse with humans, write code or craft poems with a low amount of error (ChatGPT; OpenAI, 2022). The two most common transformer architectures for NLP are visible in Figure 4.

Unlike Long Short-Term Memory networks ("LSTM"; Hochreiter and Schmidhuber, 1997), a subcategory of RNN and the previously most widely used DNN for NLP, transformers use a mechanism called *self-attention* to process input sequences in a parallelised manner. This mechanism allows them to handle longer inputs of variable length, and make both training and using them easier and faster thanks to the added opportunity of parallelisation. We can therefore define them as follow:

Definition 9. ***Transformer****: A Deep Neural Network which uses mechanisms of self-learned attention to process entire input sequences in a parallelised, permutation invariant manner.*

Specifically, self-attention refers to the model using dynamically learned attention weights to process input sequences while attending to all parts those sequences simultaneously. This results in two things: on one hand, a transformer is able to self-focus its attention on key points within the input while not looking too deeply into parts deemed less interesting; while on the other hand, it allows it to process the relation between words without being hampered by the distance between them - something that was commonly

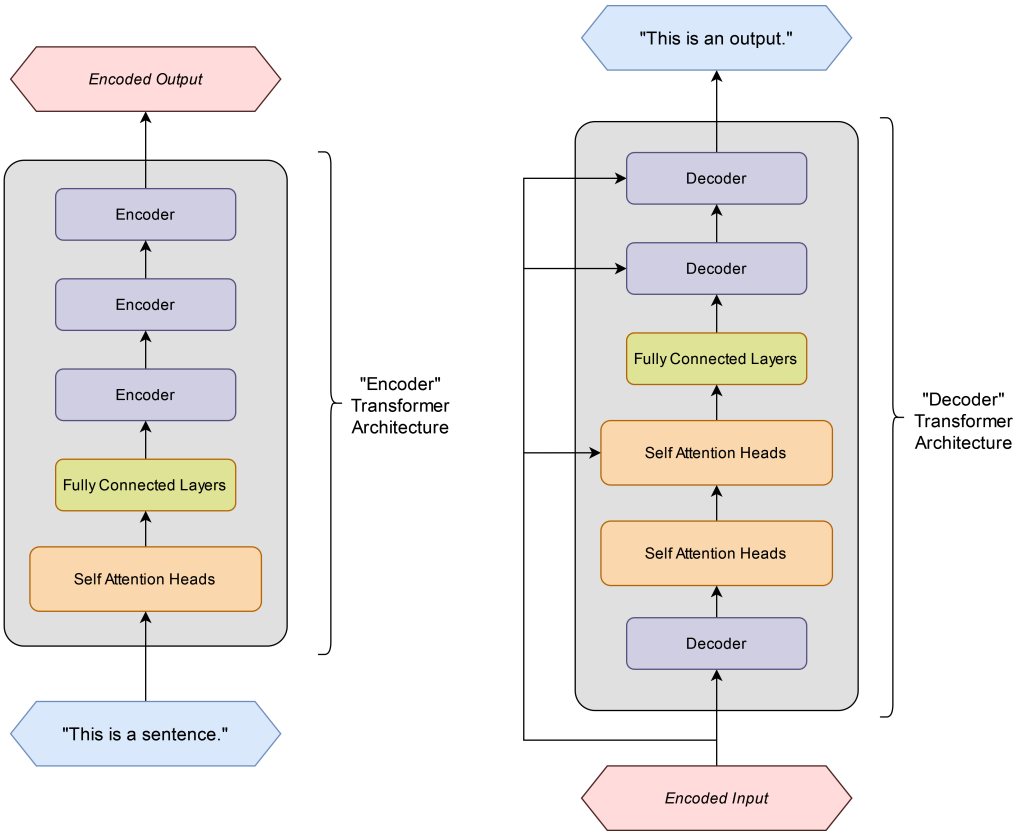

Figure 4: The two most common architectures of transformers for NLP. An encoder transforms a text into a coded output that can then be used by further neuron layers or full ANN, a decoder transforms a coded input into readable text. Both are combined in "Sequence To Sequence" tasks like Machine Translation.

hurtful to RNN because of the problem of "*vanishing*" or "*exploding*" gradients (discussed in Bengio, Frasconi, and Simard, 1993).
Doing so is highly valuable to NLP tasks in particular, as the context around words - as much before than after them - is important to better understand their meaning and role in a sentence. Their architecture also inherently scales with training data and model size - meaning that larger models and more data usually yield better results.

On the more technical side, self-attention mechanisms work by first projecting the input sequence into a higher-dimensional space using a linear transformation. The model then computes the dot product of this transformed input with a set of learned weights. Finally, those dot products are scaled via a learned scalar factor and normalized[10] to obtain attention weights, which represents the importance of each elements according to the algorithm's training.

[10]Usually, the normalization is done with a *softmax* function which transforms all weights to numbers between 0 and 1, with their sum equal to 1.

3 Theoretical Background

This chapter will summarise the literature reviewed while writing this master's thesis, providing a theoretical and historical background to the task at hand. We will first approach the more general subject of NLP, before delving into the more specific case of chiasmus detection. The chapter will then end on an additional note about linguistics, brought to our attention as we were studying the past works on chiasmi.

3.1 Natural Language Processing

3.1.1 A historical Review of NLP

3.1.1.1 The early History

The history of modern NLP[11] arguably begins in 1950, when Alan Turing described the *Turing Test* - called the *Imitation Game* in Turing's original article (Turing, 1950). This particularly well-known eponymous thought experiment describes a test through which we could test the "intelligence" of a machine, focusing on its ability to understand and communicate with humans via textual means without being noticed as a machine. Admittedly, this "test" can not be considered as a true proof of intelligence or thinking processes should a machine pass it, and has been criticised from a scientific perspective. The book "Parsing the Turing Test" (Epstein, Roberts, & Beber, 2009) is an excellent example of that, with numerous editors' notes from various sources on top of the original article showing several of its weaknesses. However, it is still the first wildly spread article evoking the real possibility of machines understanding - and communicating in - human languages.

A few years after Turing's thought experiment, on January 7th, 1954, a particularly influential event happened in New York, in the headquarters of IBM: the Georgetown-IBM Experiment on machine translation. Described and discussed in detail in Hutchins (2004), the experiment made the front page of the New York Times on January 8th, 1954 as "a public demonstration of what is believed to be the first successful use of a machine to translate meaningful texts from one language to another". The public event saw the translations of more than sixty Russian sentences into English, using around 250 words and six grammatical rules, and sparked at that time interest and hope for a general translating Machine in the coming years. This *tour de force* was the results of the work of four persons: from the Georgetown University, Leon Dostert[12] and Paul

[11] Alongside various web and academic researches, Khurana, Koli, Khatter, and Singh (2022) and Kumar (2013) were the two main sources of information for this historical overview.

[12] Leon Dostert was the person at the origin of the project, which he imagined after attending the first conference on Mechanical Translation in 1952 (Reynolds, 1952).

Garvin and from IBM, Cuthbert Hurd and Peter Sherida.
Nowadays however, the Georgetown event is often described as having be held too early in regards to the advancements of the Machine Translation field at that time, with a system fine-tuned and doctored to work on a very specific set of sentences from organic chemistry (Hutchins, 2004). It led to a frenzy of researches and funding that eventually finished on a disappointment with the ALPAC report of 1966, with its publication dramatically diminishing the funding for Machine Translation researches. It is a good example of the necessity to take advancement in the fields of NLP with caution, as good results on a specific task can not always be generalised.

Some more years after the Georgetown experiment, Noam Chomsky published his revolutionary book "*Syntactic Structures*" (Chomsky, 1957) (then followed by his complimentary work, "*Aspects of the Theory of Syntax*" (Chomsky, 1965)), in which he argues for the decoupling of the study of syntax and semantic in Linguistics. Using the two sentences "Colorless green ideas sleep furiously" and "Furiously sleep ideas green colorless" as an example, he argues that although both are semantically nonsensical, the former is grammatically (i.e. syntactically) sound. He then goes on to describe a fully formal approach to the structure of languages, arguing that it can be understood in terms of a set of rules that generate all possible sentences of a given language. In order to support his theory, Chomsky introduced several key concepts, including the idea of a "deep structure" that underlies the surface structure of a sentence, and the concept of a "transformational rule" that allows for the manipulation of deep structures to create different surface structures. *Syntactic Structures* had a major impact on the field of linguistics, and is the origin of many further NLP researches, including but not limited to the use of formal systems to understand and generate natural language.

After Chomsky came a number of additional researches using his theory of formal linguistics in the form of handcrafted rule-based systems as the core of their research. Some notable examples are ELIZA (1964) and PARRY (1972) in the field of psychotherapy, SHRDLU (1970) understanding logical relations in a simple simulated world, or LIFER / LADDER (1978) used as a natural language interface with a database about the U.S. Navy Ships. Moreover, a number of datasets were made publicly available, making natural language easier to study with the first (and one of the most known) of those dataset being the Brown corpus of Standard American English (Kučera & Francis, 1963-64).

3.1.1.2 The Advent of statistical Models

Starting from the late 1980s, change began coursing throughout the NLP community. The research focus partly shifted from manually tuned models based on Chomsky's

formal theory of language and formal logic to statistical models. Instead of basing their predictions on exhaustive, explicit grammatical rules and logical relations described and hard-written in the systems by humans, those new models used statistical methods to "learn" natural languages from existing data. One of the earliest example of such a model is presented in Benello, Mackie, and Anderson (1989), in which a simple artificial neural network of 560 neurons is used to disambiguate the syntactic category of words in sentences. Most of the models developed during this period, however, did not use ANN and preferred the more classical statistical methods described in subsection 2.2.1. This can be explained because ANN usually need more data and - more importantly - more computing power to be trained, which were rare resources at that time.

This new focus, which essentially joined the fields of Machine Learning and NLP, emerged for a few reasons (Manning, 2000). First of all, natural languages are by nature ambiguous and processing them require word knowledge that is not part of the input text itself: statistical models can thus better model those ambiguities and external word knowledge than formal grammar based models, which are rigid and mostly focused on the structure of sentences. Moreover, probabilistic models can take into account the variations of language between communities or time periods much more easily than logic, formal grammar based models. Finally, using formal grammar as the main way to process natural languages can also be problematic because the rules of grammar itself are often bent by people, be it for rhetorical purposes, regional dialects or simply because of mistakes[13].

By using ML instead of rule-based models, the research on NLP saw rapidly improving results during the 1990s and 2000s. In 1987, Sondheimer described the "rate of progress in natural language processing [to have] been disappointing to many, including [himself]" because of "overblown expectations" both from the press and the NLP community itself but change for the better was soon to happen. A good example is the field of Machine Translation, which saw such improving results (Hutchins, 2007) with the advent of SMT: Statistical Machine Translation. One of the first fully functional example is Candide, from Berger et al. (1994), a translation system from French to English based solely on statistical methods. At the time, Candide beat the previous state of the art both on *fluency* and *adequacy* of translations[14] for fully automatic translations while humans aided by its assistant mode obtained better results on both measurements than those that translated without the machine's help. Another prominent example are web search engines (Seymour, Frantsvog, & Kumar, 2011), with the introduction in 1998 by

[13] For example, the sentence "You coming?" (and other variations) is grammatically incorrect in English, but is still often used colloquially in some social and geographic circles.

[14] *Fluency* stands for the syntactic quality of the translation: is the translation a proper sentence in the target language? *Adequacy* stands for the semantic quality of the translation: is the translation adequately conveying all the meaning of the source sentence?

Google of the PageRank statistical method which allowed users to see a list of results ranked by relevance and importance of the web pages rather than a rank purely based on the number of occurrences of the search words in the target pages like its predecessors. The development of PageRank allowed users to navigate on and search the ever growing Internet much more easily and improved versions of it are nowadays used by all search engines.

3.1.1.3 The Resurgence of ANN

The transition towards classical ML models from purely rule-based models allowed NLP processes to reach new heights and finally fulfill more and more practical use cases. It was, however, not its latest development: over the last two decades, a small revolution happened in the field of NLP (and more generally, for the whole of ML researches) with the emergence of efficient and usable ANN (Kamath, Liu, & Whitaker, 2019). ANN and even more so DNN proved particularly effective at NLP because of their ability to process a large amount of data and extract statistically significant properties out of it, much more so than classical Machine Learning methods - at the cost however of more difficult and computationally intensive training. For NLP in particular, such data is often readily accessible through existing large corpora like COCA (Davies, 2008) or the aforementioned Brown Corpus (Kučera & Francis, 1963-64) and in some cases may even be collected directly from the Internet.

This neural networks resurgence started twenty years ago (Kamath et al., 2019), with the work of Bengio, Ducharme, and Vincent (2000) introducing an ANN capable of processing words into a dense vector representation, and only grew stronger since. Here follows a non-exhaustive list of notable milestones achieved thanks to ANN and DNN since Bengio et al.:

- Collobert and Weston (2008) and Collobert et al. (2011), introducing the very efficient concepts of pre-training and multitask training, showing that several NLP tasks could be achieved with better results using neural networks instead of classical ML methods.
- Mikolov, Sutskever, Chen, Corrado, and Dean (2013), proposing a highly improved computation of Bengio et al.'s word representation.
- Kalchbrenner, Grefenstette, and Blunsom (2014) which improved the capacities of locally context-sensitive NLP tasks by introducing an improved Convolutional Neural Network architecture.
- Sutskever, Vinyals, and Le (2014) which presented the concept of sequence to sequence learning using Long Short-Term Memory (a subset of Recurrent Neural Networks).

In particular, this latter concept was then introduced to machine translation softwares, dramatically improving the quality of their output and bringing them closer to the level of human specialists. Widely known examples of this are the modern version of Google Translate or DeepL.

3.1.2 Nowadays: Transformers, or Leveraging Attention Mechanisms for NLP

The progress described in the last section are not, however, the current state of the art for most NLP tasks: this would instead be the transformer architecture for deep learning, introduced by Vaswani et al. in 2017. Although this architecture still has limitations and growing pains as shown in Chernyavskiy, Ilvovsky, and Nakov (2021), it has sometimes been compared to the revolution that was ImageNet for the field of Computer Vision in 2009. The main technical points and advantages of transformers are described in Section 2.2.4.

However, the simple introduction of transformers does not entirely explain in and of itself how they became so widely and quickly used: the work of Wolf et al. (2020b) greatly helped them reach the NLP community by introducing "*Transformers*", an open-source Python library by the Hugging Face community. This library assembles, under a unified API and in a fully open way[15], almost all non-proprietary transformers architecture including its biggest names like BERT or GPT. The unified API in particular allows to easily and quickly use, train or adapt any transformer architecture without having to deal directly with PyTorch and TensorFlow[16] while still leaving the possibility of fine-tuning parameters and architecture as much as necessary to the more advanced users.

A particularly noteworthy example of transformer is the GPT (Generative Pre-trained Transformers) family of transformers from OpenAI. Building upon the work of Vaswani et al., they created several versions of transformers over the years, each offering better results than the previous ones. In particular, their chatbot ChatGPT (OpenAI, 2022) based on GPT-3 made the headlines in late 2022 for its remarkable ability to converse with people on a wide variety of topics, in an English which is almost indistinguishable from human conversations.

3.2 "The Case of Chiasmus"

Chiasmi in general, and antimetaboles more specifically, only began drawing attention from the research community a decade and a half ago. As such, the background in this

[15] Anyone in the community can propose a new transformer or rate and comment on existing architectures.

[16] The two biggest library for direct ANN building and training in Python.

subject is still relatively small with only a handful of names having made significant contributions. These researchers have however opened the gates and prepared the field for all subsequent works, and their contribution should not be understated.

3.2.1 The first Extensive Researches

The very first research we found that tried to algorithmically detect a given type of chiasmus was the work from Gawryjolek (2009), fourteen years ago. The subject of this thesis was the automated annotation and visualisation of rhetoric figures, and one such figure was the antimetabole. In a brief section of his work, Gawryjolek describes a simple algorithm (Algorithm 3.2, p. 26) to detect the most basic antimetabole: the one where two or more exact words would be repeated in an inverse order. As described by Gawryjolek himself, this algorithm has two main flaws: it "produces a lot of antimetaboles that are not necessarily important from the rhetorical point of view" (i.e. non-salient antimetabole) since it detects all inverse repetition of words, and "[it does] not look at different forms of a word, but only at repetitions of exactly the same words." which means a sentence such as Example 5 below would not be detected. Gawryjolek thus proposed that a human should go over the algorithm's results for annotation purposes for now while additional work would have to be done to make the computer detection able to separate salient antimetabole from non-salient ones.

- ***(5)** To be **kissed** by a **fool** is stupid; To be **fooled** by a **kiss** is worse.*[17]

Building upon Gawryjolek's idea of simple, deterministic algorithms then came the work of Hromada (2011) and Dubremetz (2013) (in French). Hromada proposed PERL Regular Expressions as a mean to detect chiasmus, looking for the specific case of repeating words with a middle pivot (i.e. sentence structured as follow: "W_a W_{pivot} W_b ... W_b W_{pivot} W_a") whereas Dubremetz decided instead to improve on Gawryjolek's algorithm, adding several more filters regarding punctuation and stopwords to improve the precision of the algorithm dramatically (From 2% to 72%) while keeping a similar recall.

Thus come the serie of articles written by Marie Dubremetz and Joakim Nivres six years after Gawryjolek's thesis, with the first paper being titled "*Rhetorical Figure Detection: the Case of Chiasmus*"[18] (Dubremetz & Nivre, 2015). In this first work of a serie of three, they studied how manually fine-tuned statistical methods (they could not use proper ML techniques at that time due to the lack of data) could be used to sep-

[17] Ambrose Redmoon.
[18] Part of this title was used as title for this section of the thesis to honor their work.

arate salient antimetaboles from uninteresting ones[19]. Moreover, they argued that the saliency of chiasmi should be seen as a ranking problem instead of classification one: according to them, we cannot say with full certainty that a criss-cross pattern is or is not a salient chiasmus but simply that a chiasmus is more salient than some others. They thus decided for a two steps process as their methodology: first, an algorithm similar to Gawryjolek's extract any possible antimetabole candidates by looking for criss-cross patterns of lemmata, then those candidates are ranked by a manually tuned standard linear ranking model using shallow features. Doing so, they obtained better results than the previous state of the art by Hromada by a few points of percentage both in precision and recall. Furthermore, they also used their new model to detect a number of Chiasmi in the *Europarl* dataset[20], offering more data for future works.

One year after, they wrote a new article on the same subject (Dubremetz & Nivre, 2016). While building upon the previous work by reusing the same kind of ranking model and general methodology, this paper presented one major improvement: the addition of a new set of syntactically deep features based on part-of-speech (PoS) tagging, adding depth to the shallow model of 2015. This new model saw a very noticeable improvement in its result: using the average precision metric and compared to the state of the art from Dubremetz and Nivre (2015), its results were improved by 25 points (68% vs 43%) on the Europarl corpus and by 17 points (70% vs 53%) on the complete anthology of the Sherlock Holmes serie by Arthur Conan Doyle, a literary genre which it never encountered before. However, they note that due to the very low number of results, those improvements can not be taken as an absolute proof of a better models but may also be statistical flukes. Finally, like with their previous article, they compiled all new chiasmi found with their model in a single place, paving the way with enough data for the first ML methods soon after.

The final article in this academic serie was "*Machine Learning for Rhetorical Figure Detection: More Chiasmus with Less Annotation*" (Dubremetz & Nivre, 2017). As its title suggests, the main improvement they proposed with this work is the introduction of ML methods to tune their ranking model - a feat made possible thanks to the data they collected with their previous two experiments. To be precise, they used a binary logistic regression classifier with the same features as Dubremetz and Nivre (2016) and two fold cross-validation (a Kernel SVM model was also tried and yielded similar results to this classifier) and trained their model with the 31 instances annotated as "true" from the previous works considered as positive examples, while keeping the same candidate

[19] Although their work only focus on antimetaboles, they refer to them with the broader term chiasmus instead. Moreover, what we call "salient antimetabole" is referred in their articles as "true chiasmus", with the opposite being "random criss-cross patterns".

[20] A corpus of debates from the European Parliament, `https://www.statmt.org/europarl/`.

Feature	Description
Basic	
#punct	Number of hard punctuation marks and parentheses in C_{ab} and C_{ba}
#softPunct	Number of commas in C_{ab} and C_{ba}
#centralPunct	Number of hard punctuation marks and parentheses in C_{bb}
isInStopListA	W_a is a stopword
isInStopListB	W_b is a stopword
#mainRep	Number of additional repetitions of W_a or W_b
Size	
#diffSize	Difference in number of tokens between C_{ab} and C_{ba}
#toksInBC	Position of W'$_a$ minus position of W_b
Similarity	
exactMatch	True if C_{ab} and C_{ba} are identical
#sameTok	Number of identical lemmatized tokens in C_{ab} and in C_{ba}
simScore	#sameTok but normalised
#sameBigram	Number of bigrams that are identical in C_{ab} and C_{ba}
#sameTrigram	Number of trigrams that are identical in C_{ab} and C_{ba}
#sameCont	Number of tokens that are identical in C_{Left} and C_{bb}
Lexical clues	
hasConj	True if C_{bb} contains one of the conjunctions 'and', 'as', 'because', 'for', 'yet', 'nor', 'so', 'or', 'but'
hasNeg	True if the chiasmus candidate contains one of the negative words 'no', 'not', 'never', 'nothing'
hasTo	True if the expression "from ... to" appears in the chiasmus candidate or 'to' or 'into' are repeated in C_{ab} and C_{ba}
Syntactic Features	
sameTag	True if W_a W_b W'_b W'_a words have same PoS-Tag.
#sameDepW_a W'_b	Number of incoming dependency types shared by W_a and W'_b.
#sameDepW_b W'_a	Same but for W_b and W'_a
#sameDepW_a W'_a	Same but for W_a and W'_a
#sameDepW_b W'_b	Same but for W_b and W'_b

Figure 5: Dubremetz and Nivre's textual and syntactic features used in the models presented in their 2016 and 2017 articles, from Dubremetz and Nivre (2017).

extractor as Dubremetz and Nivre (2015). Using Machine Learning, they thus improved the average precision by 3.1 points (70.8% vs 67.7%) over their previous best result from 2016.

The final set of features, used in their two last articles (Dubremetz & Nivre, 2016, 2017) can be seen in Figure 5. Their first article (Dubremetz & Nivre, 2015) used the same list of features except for the "Syntactic Features" category.

3.2.2 Using Semantic Features: the current State of the Art in Chiasmus Detection

The latest improvement to chiasmi detection came from Schneider et al. (2021): although heavily based on Dubremetz and Nivre (2017), their work was different on few key points. First of all, and most notably, they broadened the scope of the detection by focusing not only on antimetaboles but on general semantic chiasmi: in practice, they

Feature	Description
Lexical Features	
$EqualLemmaW_aW'_a$	True if the lemmata of W_a and W'_a are equal.
$EqualLemmaW_aW_b$	Same for W_a and W_b.
$EqualLemmaW_aW'_b$	Same for W_a and W'_b.
$EqualLemmaW_bW'_a$	Same for W_b and W'_a.
$EqualLemmaW_bW'_b$	Same for W_b and W'_b.
$EqualLemmaW'_aW'_b$	Same for W'_a and W'_b.
Embedding Features	
$CosineSimilarityW_aW'_a$	The cosine similarity between W_a and W'_a.
$CosineSimilarityW_aW_b$	Same between W_a and W_b.
$CosineSimilarityW_aW'_b$	Same between W_a and W'_b.
$CosineSimilarityW_bW'_a$	Same between W_b and W'_a.
$CosineSimilarityW_bW'_b$	Same between W_b and W'_b.
$CosineSimilarityW'_aW'_b$	Same between W'_a and W'_b.

Figure 6: The additional features from Schneider et al. (2021), added on top of the previously existing features presented from Dubremetz and Nivre (2017).

aimed to detect any rhetorically salient criss-cross patterns of semantically connected words (be it because similar or opposite meanings) and not only inverse repetitions of lemmata. To support this shift in goals, they had to change the way the extraction of candidates was done: they therefore chose to look for criss-cross patterns of PoS tags instead of those of lemmata. Finally, they added two sets of features to the features from Dubremetz and Nivre: a set of lexical features to compensate for the loss of the intrinsic equality of lemmata, and a set of word embedding features to model the semantic relations between the clauses of their chiasmi candidates. Those additional features can be seen in Figure 6.

Apart from those changes, their process and models stay the same: they first extract any possible chiasmi candidates before feeding them to their candidate ranking model - here again a logistic regression classifier. In particular, their model was trained and validated using five fold cross-validation twice: once on German data, using a fully annotated compilation of four different works from the author Friedrich Schiller and once on English data, using the dataset from Dubremetz and Nivre (2017).

Using these new methods, they were able to improve upon the work of Dubremetz and Nivre (2017) (which will be called "Dubremetz's method" from hereon). Figure 7 shows that: we can see that their method had similar results as Dubremetz and Nivre's on the English dataset made only of antimetaboles, but that they beat it on the German dataset on all fronts. Even for antimetabole, their best results was an average precision of 49% whereas Dubremetz's model only has an average precision of 21%. Interestingly, those results tend to show that using lexical features is either neutral or damaging to

	Schiller dramas			Dubremetz data
	Antimetaboles	Chiasmi	Combined	Antimetaboles
(baseline) D	0.21 ± 0.18	0.15 ± 0.07	0.17 ± 0.04	$\mathbf{0.73 \pm 0.24}$
L	0.07 ± 0.08	0.01 ± 0.00	0.02 ± 0.00	0.06 ± 0.01
E	0.10 ± 0.07	0.06 ± 0.03	0.09 ± 0.07	0.25 ± 0.13
LE	0.11 ± 0.06	0.05 ± 0.02	0.08 ± 0.04	0.25 ± 0.13
DL	$\mathbf{0.49 \pm 0.32}$	0.14 ± 0.09	0.22 ± 0.07	0.72 ± 0.24
DE	0.48 ± 0.30	$\mathbf{0.23 \pm 0.13}$	$\mathbf{0.28 \pm 0.01}$	$\mathbf{0.73 \pm 0.24}$
DLE	0.48 ± 0.30	0.19 ± 0.09	$\mathbf{0.28 \pm 0.08}$	$\mathbf{0.73 \pm 0.24}$

Figure 7: Results from Schneider et al. (2021) on four Schiller dramas and on Dubremetz and Nivre's dataset.
D = Dubremetz features ; L = Lexical Features ; E = Embedding features.

the performances of their method, as the model with Dubremetz's and Embedding features was either better (to detect non-antimetabole chiasmi) or almost equal (to detect antimetabole or all chiasmi combined) compared to the models with lexical features. The differences are however particularly small and well within the margin of error, and nothing conclusive can be extracted from this specificity.

Finally, it is important to note that while their method offers much better results in those experiments than the one from Dubremetz and Nivre (2017), a key point must be considered: this study was conduced mostly with German texts, whereas Dubremetz and Nivre conduced theirs in English and while the comparison is interesting, this difference needs to be taken into account when evaluating the final results. Moreover, their method did miss some prototypically salient before even reaching the ranking phase: because it was founded on inverse repetitions of PoS tags, their candidate extraction phase missed chiasmi and antimetabole were the words did not reverse their part of speech. This was not the case with Dubremetz and Nivre's extraction, as it focused only on lemmata. Such an example of a missed chiasmi can be seen in Example 5, where the PoS tags of the chiastic clauses show a parallelism instead: "*Noun ... Verb ... Noun ... Verb*".

3.3 An Addendum on Linguistics

Although this thesis focuses mainly on the Computer part of "Computer Linguistics", the Linguistics part should not be entirely ignored. This section thus focuses on a few researches and arguments from linguistic works that can help better define the subject of rhetorical figures as a NLP problem.

In Harris et al. (2018), the authors combines a detailed meta-study on the researches on rhetorical figures in the field of computer linguistics and a proposition for a generalist annotation scheme of such rhetorical figures. They also adequately argue that the

question, when working with rhetorical figures such as antimetaboles, is not so much about detecting true antimetabole from false ones but rather to detect those which have a rhetorical purpose from those which do not and are only incidentally chiastic[21]. This is the reason why this thesis is titled "Detecting Salient Antimetaboles" and not simply "Detecting Antimetaboles": the second one would need an algorithm only as simple as Gawryjolek (2009)'s.

It is also interesting to note that the researches that have been done on rhetorical figures from a linguistic point of view can be useful for those trying to solve NLP problems, especially those trying to comprehensively organise and define them. A good example of such is Mitrović, O'Reilly, Mladenović, and Handschuh (2017), in which the authors propose an ontology of rhetorical figures. This ontology could then be useful when using machine to detect those figures: a software could look at any given text, compare it to the rules present in the ontology, and mark as a possible rhetorical device those that follow all the necessary rules.

A last but quite important point stemming from linguistic researches, present in both Harris et al. and Mitrović et al. (2017), is the argument that a complete work on the subject should not only try to automatically detect rhetorically salient figures but also train or tune models to understand their rhetorical purpose efficiently. We consider it a particularly important step to take for the field of Computer Linguistics, but we were not able to take it ourselves within this thesis due to a lack of time.
However, this point appeared to us important enough to share it with any computational linguists reading this work, in order to prompt them to think about this important "question of functions", a question whose answer could further the understanding of natural languages by machines.

[21] This idea extends to most rhetorical figures: for example, an anaphora is always an anaphora but not all of them have a rhetorical purpose.

4 Methodology

As stated before, this thesis's goal was to investigate the usefulness of Deep Learning to detect antimetabole. However, several intermediary steps were taken to reach this point and this chapter therefore aims to describe the methodology used throughout the last year for this work. We will begin by describing the preliminary researches which we conducted, before talking about a newly built chiasmus candidate extraction tool and finally the most recent developments: proper data gathering and collation, and the usage of transfer learning to train a transformer model.

It is important to note that we decided to focus on the detection of salient antimetaboles only in the later part of the thesis, and a lot of the work during the first half of it has been done with the broader task of detecting salient chiasmi[22] in mind. This explains some of the decisions we took during our intermediary work.

4.1 Preliminary Researches

The early stages of this thesis were conducted with Yohan Meyer and were focused most of all on academic researches and understanding of the general subject: we thus began with the three articles from Dubremetz and Nivre described in Section 3.2.1, as they are the foundation of the modern researches on chiasmi detection using statistical models and machine learning. More specifically, our efforts during this period were threefold:

1. To understand Dubremetz and Nivre's work to the best of our abilities and therefore spent time learning about the techniques they used and studying their results year on year.
2. To emulate the experimental part of their latest article to use it as a possible basis for future work, i.e. our work.
3. To collect the data they used to form the foundation of a "Chiasmus Dataset".

The first goal was mostly done through personal academic researches, and was fulfilled without any major hurdle. The second and third one needed inputs from Dubremetz and Nivre, and we hence quickly resolved to contact them to ask for both their code and their dataset, a request to which they promptly answered positively. We were thus able to save their dataset as a foundation for more data gathering.

However, even after obtaining the code they used in their researches, we were unable to properly run it: the code base had problems running in a modern python environment, since it was written in Python 2. This version of the language was indeed sunset

[22] More precisely, both semantic chiasmi and lexical chiasmi.

on January 1st, 2020 which makes it hard to properly work on softwares written with it. Consequently, we decided not to try further and instead focus on other sources for possible implementations.

4.2 Building an Extraction Software for Chiasmi Candidates

Following our work focused on the Dubremetz and Nivre's articles, we turned towards Schneider et al. (2021) as inspiration. As explained in Section 3.2.2, we consider it to be the current state of the art in chiasmus detection despite some weaknesses in their methods. Recognising that detecting candidates using PoS tagging misses some important chiasmi, we then decided to build our own candidate extraction tool using ideas both from Dubremetz and Nivre and Schneider et al..

First of all, we followed their example and decided to look for candidates using a sliding window of thirty words. Almost all chiasmi are built with clauses within thirty words of each other, and those who do not fit within this limitation then tend to have a wide distance between their clauses (for example, a chiasmi whose first half is the first sentence of a book and the second half, its last sentence). A theoretically perfect algorithm would be able to look at any and all possible word combinations of its input, but this 30-words window has been shown to be a good trade-off between computational power and high detection rate by previous researches. Furthermore, we also needed to tokenise the document, associating to each words their lemmas, PoS and positions. To do so, we chose to use the Stanza pipeline[23] (Qi et al., 2020) because of its completeness, effectiveness and efficiency when working on English documents.

With the sliding window fixed to thirty and the document properly parsed, we turned ourselves to the detection process itself. As mentioned above, we could not solely use PoS inversion but neither could we simply use lemmata inversion like Dubremetz and Nivre, as we were looking for general chiasmi at this stage of the thesis. Our first decision was therefore to let go of PoS tagging as a detection method because the very high number of candidates it produces would highly slow down the entire detection pipeline. We then decided to use a combined approach: on one hand, our tool would look for words with the same lemmata like Dubremetz and Nivre to make sure we would not miss any possible antimetabole, while on the other hand, we would use word embedding to look for combination of either synonyms or antonyms as possible candidates.

Word embedding is a method that is used to project words as vectors in a high-dimensional space using a DNN. In this space, the angle and distance between the words' vectors is

[23] `https://stanfordnlp.github.io/stanza/tokenize.html`

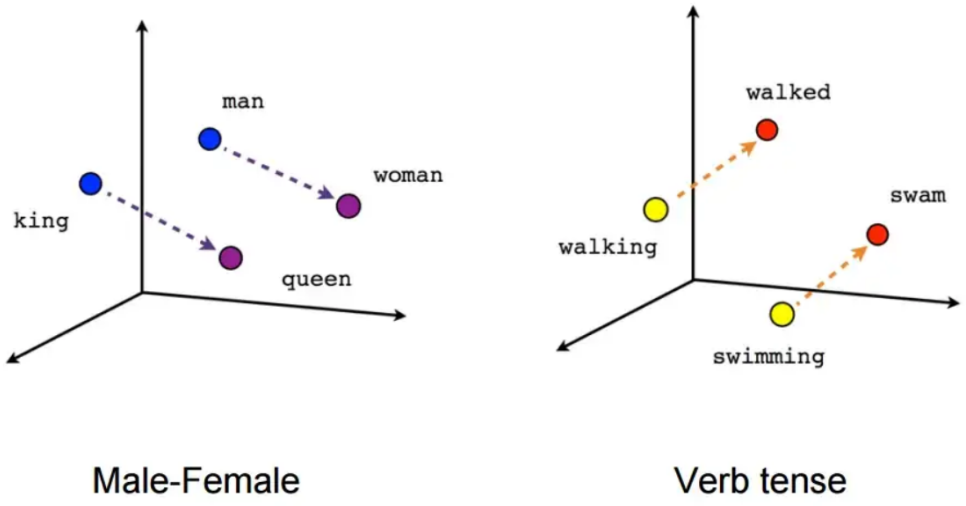

Figure 8: A simplified visualisation of the core principles of word embedding: the distance between words is supposed to be representative of their distance in actual meaning. From Ruizendaal (2017).

supposed to represent - to a degree - the difference between their actual meanings (see Figure 8 for a simplified visualisation of this property).
A particularly useful computation when manipulating word embedding is the "cosine similarity": it takes two word embeddings as an input and output a number between -1 and 1, depending on the similarity of their angles. A similarity score close to 1 means the vectors are almost overlapping and thus that the words have a similar meaning according to the embedding model, whereas a score close to -1 means the vectors are opposite to each other and have an equally opposite meaning. In our candidate extraction algorithm, we therefore decided to consider any pair of words whose cosine similarity scored above a certain threshold (synonyms) or below the same threshold in the negatives (antonyms) as a possible chiastic pair for chiasmi candidates.

In our case, we opted for *GloVe*[24] in its Common Crawl (48B) version as our embedding software. We decided to use it because of its high result in other NLP tasks while being easy to access and use directly thanks to the existence of several different pretrained versions of it.

Algorithm 1 thus shows a simplified version of the core loop of our tool. Our actual implementation[25] is however more precise in one way that would have been too long to show in a pseudocode algorithm: it also looks for *nested chiasmus*. Nested chiasmus are the name we chose for chiastic figures with more than two pairs of words. An example

[24] `https://nlp.stanford.edu/projects/glove/`
[25] Available at `https://github.com/YohanMeyer/ChiasmusExtractor`.

Algorithm 1 candidates_extraction(f: Tokenised File)

```
match_table: List of word pairs;
candidates: List of chiasmi candidates;
current_window: List of words; # Already containing the first thirty words of the document.

for each new_word in f do
    if not is_punctuation_or_stopword(new_word) then
        for each old_word in current_window do
            if old_word.lemma = new_word.lemma
               or cos_similarity(old_word.emb, new_word.emb) > THRESHOLD
               or cos_similarity(old_word.emb, new_word.emb) < −THRESHOLD
            then
                current_match ← (old_word, new_word);
                for each old_match in old_matches do
                    if old_match.w1.position > current_match.w1.position then
                        Append (current_match, old_matches) to candidates;
                    end if
                end for
                Append current_match to match_table;
            end if
        end for
    end if
    Append new_word to current_window;
    Pop the oldest word from current_window;
end for
return candidates;
```

can be seen with Example 6 which is an antimetabole made of three different pairs of words. One could argue that such candidates could be detected simply by detecting the various smaller candidates that make them, but doing so would not take into account the full rhetorical potency of those structures as such candidates would be considered outside of their complete picture. By extracting them as a whole, it allows the processes depending on the results of this extraction tool to take nested chiasmi into account if they wish so.

- ***(6)*** ***Baby*** , ***you*** *'ve been so distant from me* ***lately*** *... And* ***lately*** *, don't even want to call* ***you*** ***baby*** .[26]

4.3 Constructing a Dataset

When the work on this thesis began, one of the main goal that was set was to improve the quantity of the data available for chiasmus detection. To do so, we explored two

[26] Selena Gomez.

different methods: manually compiling chiasmi from various sources and using a combination of the candidate extraction tool described in the previous section and the current state-of-the-art in chiasmi detection to extract more chiasmi from new sources.

4.3.1 Manually Gathering New Examples

To achieve the first part, we gathered chiasmi examples from previous articles in the field (notably from articles published by Dubremetz and Nivre, Schneider et al., Harris et al. and Gawryjolek), various internet websites - often dedicated to rhetorical figures at large or chiasmi in particular - and one specific book: "*Never let a fool kiss you or a kiss fool you*", by Dr. Mardy Grothe (2014). This book contains a large collection of chiasmi and allowed us to collect a large amount of chiasmi of several types, from antimetaboles to chiasmi playing on the pronunciation of words[27] and after gathering all its examples of chiasmi and filtering the uninteresting ones, it totaled to ***507 examples*** of chiasmi (either semantic chiasmi, phonetic chiasmi or lexical chiasmi i.e. antimetaboles).

4.3.2 Using a Semi-automated Pipeline to Extract more Examples from Novel Sources

The other idea we had to expand the number of positive examples of chiasmi was to use a similar method to that of Dubremetz and Nivre (2015, 2016): using existing models and processes to extract highly ranked candidates from new corpora. To do so, we imagined a new architecture for a semi-automated pipeline: first, we would extract candidates from new corpora and literary sources; then, we would feed those candidates to the model from Schneider et al. (2021) to rank them; the best ranked candidates would then be manually annotated using the Doccano annotation tool (Nakayama, Kubo, Kamura, Taniguchi, & Liang, 2018), giving from this point an already usable list of positive (and negative) examples of chiasmi; finally, the positive examples of chiasmi would be combined with the original input texts according to Harris et al. (2018)'s XML annotation scheme, allowing for more context around the chiasmi themselves for the processes that might use it. This whole pipeline can be visualised on Figure 9.

However, we were never able to fully bring this idea to life for three reasons. Two of those were linked to our extraction tool: at the time of writing, its current version is particularly slow due to a weakly optimised codebase and its results regarding anything but pure antimetabole are particularly low (as will be further described and explained in Chapter 5). The third reason is that we could access the model described in Schneider

[27] "A magician pulls rabbits out of hats. An experimental psychologist pulls habits out of rats" (*Never let a fool kiss you or a kiss fool you*, Dr. Mardy Grothe (2014)) is a good example of such a chiasmus on sounds rather than lemmata or meanings. We decided to call those *phonetic chiasmi*.

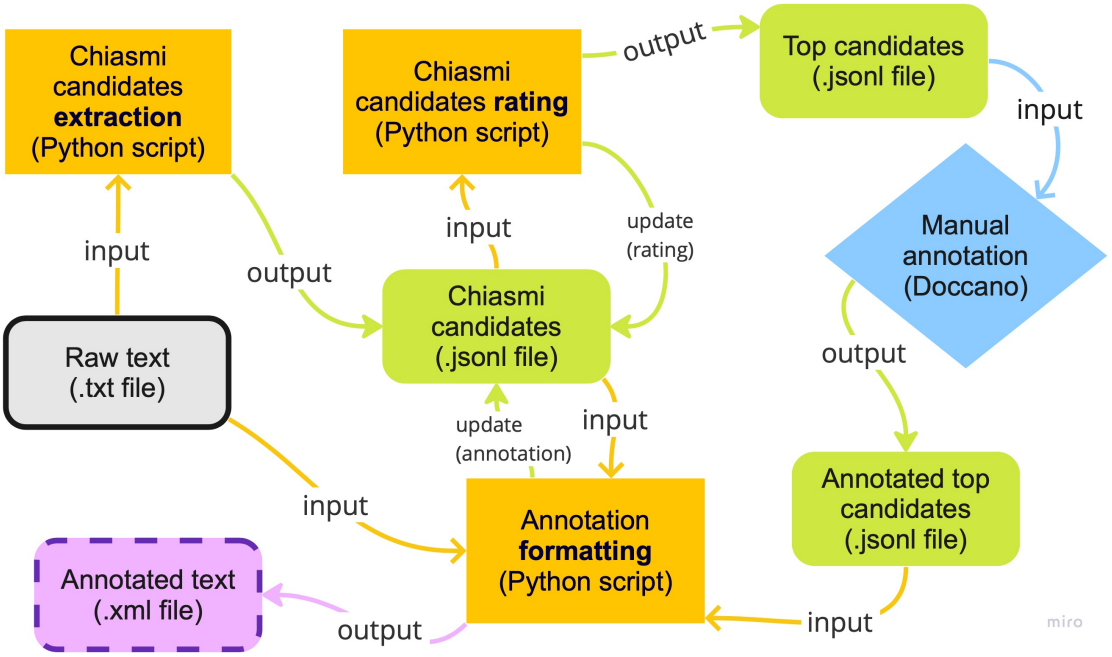

Figure 9: The imagined chiasmi annotation pipeline to detect and annotate chiasmi from novel sources (Meyer, 2023).

et al. (2021) relatively late in respect to the conduct of the whole thesis and that it would have needed some adjustment to properly work on English texts. We still deeply encourage future researches to look more into this possibility to highly enhance the existing datasets.

We however still used our candidate extraction tool to extract a heap of antimetabole candidates from a subset of the COCA corpus (Davies, 2008). As did Dubremetz and Nivre, we consider those unknown and unannotated extracts to be negative by default, because of the very low number of antimetaboles among all possible criss-cross patterns in any natural language pieces.

4.3.3 The Resulting Corpus

Despite not being able to use the second method as we first wished, we were still able to collect a vast number of chiasmi. Our corpus, fully available online[28], contains ***763*** salient chiasmi, including ***659*** antimetaboles, ***98*** semantic chiasmi and ***9*** phonetic chiasmi. Moreover, it also contains ***2720*** non-salient antimetaboles text extract and ***1388*** random sentences or pair of sentences that were not analysed for any chiastic repetition. This totals to ***3463*** examples of salient and non-salient chiasmi, and a total number of examples of ***4851***.

[28] `https://github.com/Droniil/ChiasmusDatasets` in the data subfolder.

The previously most complete English public corpus regarding chiasmi, collected by Dubremetz and Nivre (2016), contained 21 positive examples of antimetaboles against our 659 positive examples of antimetaboles. We have hence improved the previous state of the art in this regard by a factor of slightly over 30, which in turn opens up a number of possibilities for future researches on the subject - in particular those using more data-hungry ML methods like Deep Learning.

4.4 Using Deep Learning to Detect Antimetaboles

With a large collection of data now available, the next step of the thesis was to put it to use. Before any proper work, two important decisions had to be taken. The first was motivated by a review of the literature presented in Section 3.2 which only contains classical statistical and ML methods to detect chiasmi and thus made us decide to use a novel approach for this problem: the recent but particularly effective transformer technology, enhanced by transfer learning. This choice was made possible in part thanks to the heap of data we collected, as transformers are much more data hungry than the classical ML methods used by Dubremetz and Nivre (2017) and Schneider et al. (2021). The second decision was taken in tandem with the first one: because of this entirely novel approach and because of the high amount of data transformers need to be trained, we decided to focus this thesis's work entirely on antimetaboles instead of looking for chiasmi at large.

4.4.1 Choosing the models

After choosing to use transformers came the question of which specific model to use. Each individual models have strength and weaknesses, coming both from the details of their architecture and the quality and quantity of data they were pretrained on. As of November 2022, the three main transformer architectures available to the public are GPTNeo[29] (Black, Leo, Wang, Leahy, & Biderman, 2021) building upon OpenAI's GPT2 and taking inspirations from their GPT3 architecture; Bloom[30] (Scao et al., 2022), similar in architecture to GPT3 but created in an open way by a community of researchers from the BigScience workshop and trained on a multilingual dataset; and BERT[31] (Kenton & Toutanova, 2019), the first widely known transformer architecture.

As the GPT architecture has a large history of effectiveness for several NLP tasks, we decided to use GPTNeo for our initial experiments, before expanding a series of experiments to the more recent Bloom architecture in order to compare both on several tasks

[29]Documentation page: `https://huggingface.co/docs/transformers/model_doc/gpt_neo`
[30]Documentation page: `https://huggingface.co/docs/transformers/model_doc/bloom`
[31]Documentation page: `https://huggingface.co/docs/transformers/model_doc/bert`

related to the detection of salient antimetaboles. Another important choice we had to make was the size of the models: the more parameters a model has, the better its results would usually be at the cost of a more data and computing resources hungry training process: in their most common versions, GPTNeo and Bloom respectively have 1.3 billions ("GPTNeo 1.3B") parameters and 176 billions ("Bloom 176B") parameters. Our first experiments thus tried to use GPTNeo 1.3B but the architecture was too heavy to be trained with the available computing resources and data, and we downgraded to the much smaller 160 millions parameters version: ***"GPTNeo 125M"***. Knowing the troubles we had to train GPTNeo 1.3B, we immediately decided to use the smallest available version of Bloom with 560 millions parameters: ***"Bloom 560M"***.

4.4.2 Choosing and Preparing the Data

The models now chosen, the next decision was that of the datasets. The first decision we took was to use all the salient antimetaboles from the dataset described in Section 4.3.3 as positive examples, except for the antimetabole from Dubremetz and Nivre (2016)'s appendices. This decision was taken so that our models could be tested against those 21 specific examples and its performances better compared to that of Dubremetz and Nivre's models. As negative examples, we first wanted to use the non-salient antimetaboles extracted by our candidate extraction tool. However, those extracts were not proper sentences as the tool would often cut sentences in the middle while extracting candidates and were thus too different from the positive examples to be properly used as negative examples. We therefore chose our initial set of negative examples to be random sentences and pair of sentences extracted from three different corpora from the data.world website: abstracts from articles prepublished on Arxiv (Friedman, 2020), English jokes from various sources[32] (Pungas, 2017) and UN Speeches (Malina, 2017).

This data was then shuffled before being split into training, validation and test subsets. We chose an 80% training, 10% validation and 10% testing split, as we had a large enough amount of data to allow for a smaller split for validation and test purposes whereas the training process of transformers wants for as much data as possible. Once the data was split, the next step was to preprocess it to make it digest for the models with the most important part being to tokenise and pad it accordingly. This tokenisation process converts one single text input into a sequence of "tokens", i.e. integer values, with each token usually representing one word or punctuation symbol in the original text. Figure 10 is a visual representation of the steps taken to tokenise a text into proper input for transformer models.

[32]To be precise, the jokes were scratched from `www.reddit.com/r/jokes/`, www.stupidstuff.org/jokes/ (nowadays unavailable) and `www.wocka.com`.

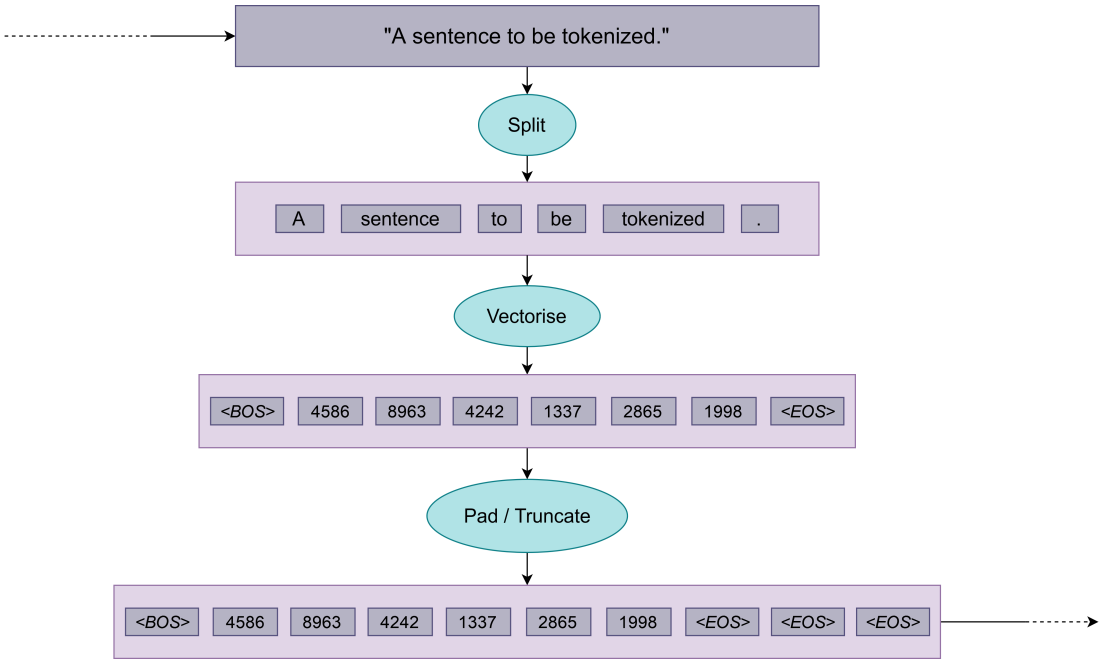

Figure 10: A visual explanation of the tokenisation process which transforms raw text into inputs usable by transformer models. *<BOS>* and *<EOS>* respectively mean "Beginning Of Sentence" and "End Of Sentence" and are used to signal the beginning and end of a sequence.

4.4.3 Setting up the training and testing processes

The training and testing pipeline of our models was setup in a Google Colab notebook[33] using cloud GPU computing resources. For all experiments, the training was done over three epochs with a batch size of 8, gradient optimisation happening every 4 batches and a mixed-precision training process. The three-epochs-training was motivated by the fast convergence of validation loss, while all other parameters were motivated by the optimisation of the process in respect to the computing resources we had. With these parameters and the data described above, GPTNeo 125M took around 3 minutes to train. The model was finally validated in two steps: first on the 10 remaining percent from the dataset, then on the 21 antimetaboles from Dubremetz and Nivre (2016).

All the models were saved after their use and are freely available online for test, training or comparison purposes[34].

4.4.4 Additional experiments

After the initial set of trials, we opted to do a few additional experiments (the codebase for those is on the same Google Colab notebook as the pipeline described above). These

[33]This notebook is available here: `https://colab.research.google.com/drive/1pP_y27gBhjB7wvdxtwGz7BCC8lrrtGCK?usp=sharing`

[34]The models are available here: `https://drive.google.com/drive/folders/1yWvkeF3p4Cgq1A3EamZ50FYrHBP4eDoJ?usp=sharing`

Algorithm 2 full_text_experiments(*novel*: Raw text, *model*: Classification model)

```
parsed_novel ← tokenize(novel);
candidates ← candidates_extraction(parsed_novel);

positive_candidates ← [];
for each candidate in candidates do
    if model(candidate).label = "Antimetabole" then
        Append candidate to positive_candidates;
    end if
end for

true_positives, false_positives ← [], [];
for each positive_candidate in positive_candidates do
    if human_annotation(positive_candidate).label = "Antimetabole" then
        Append positive_candidate to true_positives;
    else
        Append positive_candidate to false_positives;
    end if
end for

precision ← true_positives.length / positive_candidate.length;
return true_positives, false_positives, precision;
```

were divided into three parts:

- First, we used our newly trained GPTNeo 125M model to look for antimetabole in the full text of "*Frankenstein, or the modern Prometheus*" by Mary Shelley[35]. This was done in order to evaluate the results of our model on out-of-genre data.
- We then used the additional data (positive and negative) from this "Frankenstein experiment" to train a new GPTNeo 125M model from scratch. It was then tested both on the regular test data described in the previous section and on the full text of "*Dracula*" by Bram Stocker[36]. This experiment would allow us to evaluate if the added data from *Frankenstein* would improve the results of GPTNeo 125M on another literary novel.
- Finally, we trained a Bloom 560M model using the original dataset and the additional data from the "Frankenstein experiment" and tested it on the regular testing data and "Dracula"'s full text, in order to compare its performance to the GPTNeo 125M from the previous experiment.

As our transformers are waiting for sentences, we had to preprocess the full texts of both *Frankenstein* and *Dracula* into usable extracts. Consequently, we first parsed them into

[35]Full text available at `https://www.gutenberg.org/files/84/84-h/84-h.htm`.
[36]Full text available at `https://www.gutenberg.org/files/345/345-h/345-h.htm`.

sentences and lemmata using the Stanza tokenizer. We then used an algorithm similar to Algorithm 1 to look for antimetabole candidates, before feeding them to our models. Finally, the positive results of that classification were manually annotated and metrics were compiled. Algorithm 2 is a pseudo-code visualisation of this process.

It is important to note that the candidate extraction algorithm we used in this process differs from Algorithm 1 in a three notable ways: we did not use embedding at all and only looked for inverse repetitions of lemmata, we limited the search to take place only within one sentence at a time, and we outputed full sentences instead of text extracts centered on the chiasmi candidates. The reasoning behind those decisions is that our models were trained to look for antimetaboles on inputs formatted as proper sentences (or more rarely a few sentences), and we wished to obtain similar inputs during those additional experiments. Our base candidate extraction algorithm detected 576 sentences with a chiastic structure in *Frankenstein* and 1426 within *Dracula*.

5 Results

This chapter will be dedicated to the presentation of the results obtained by the various experiments described in the previous chapter. We will first talk about the candidate extraction tools we developed, before focusing on the effectiveness of the transformer models for the task. The last part will focus on the final dataset gathered throughout the thesis, including additional positive and negative examples from the "Frankenstein" and "Dracula" experiments.

5.1 Automatic Candidate Extraction

As described above, we created two different chiasmi candidates tools. The first one, presented in Section 4.2, aimed to look for both antimetaboles and lexical chiasmi through the use of embedding and to be used within a larger pipeline for annotation purposes. The second one was described in Section 4.4.4 and was much simpler, simply looking for sentences with at least one antimetabole pattern in a given text.

As those tools are made with the purpose of parsing the inputs and weeding out any impossible candidates, their precision in regards to salient chiasmi is not particularly important: any false positive would be processed by the models trained to detect salient chiasmi, and categorized consequently. Their recall, however, is extremely important: any chiasmi that is not detected during the initial process will not be categorized by the models and hence entirely ignored.

In the case of antimetabole, this recall is easy to theorise: as long as the antimetabole is not made of stopwords[37], over a window longer than thirty tokens or, in the case of the simplified tool over several sentences, then almost all salient antimetabole should be detected by the tools. However, even the most effective lemmatisation techniques have mishaps in practice, thus attributing the wrong lemmata to words with this problem preventing the detection of up to 10% of the antimetaboles of our whole dataset - although highly dependent of the files fed to it. The recall of our tool regarding semantic chiasmi is even harder to approach: our main idea was to detect them using word embedding and consider as a chiasmi any criss-cross patterns of words whose cosine similarity are above (or below the negative equivalent of) a lax threshold, and this method is subject to high performance variability depending on the embedding and comparison methods used.

To measure the general performance of our main candidate extraction tool, we launched it without and with the embedding detection method (using a 0.75 cosine similarity

[37] With our limited list of stopwords compared to even Dubremetz and Nivre (2017), very few antimetaboles would not be detected. One notable exception however still is one of the most known antimetabole: *"One for all, all for one"*.

Method	Candidates	Salient antimetaboles	Salient semantic chiasmi
Lemmata only	43	11 *(100%)*	0 *(0%)*
Lemmata + Embedding	279	11 *(100%)*	0 *(0%)*

Figure 11: The number of candidates, salient antimetaboles and salient semantic chiasmi extracted when using the chiasmi candidate extraction tool described in Section 4.2. The input text consisted of 11 salient antimetaboles and 6 salient semantic chiasmi.

threshold) on a small text file composed of 17 examples of salient chiasmi, with 11 salient antimetaboles and 6 salient chiasmi. The results of this experiment are available in Figure 11: as very clearly shown, our tool is entirely unable to detect any of the six semantic chiasmi even with a quite lax 0.75 threshold. In the meantime, it adds 236 examples of nonsalient chiasmi to to the candidate pool, all of which we know are negative in this case. Using embedding to detect salient chiasmi would thus mostly slow down the whole pipeline while bringing barely any positive results, if any.

- ***(7)*** ***He*** *went to the* ***country*** *, to the* ***town*** *went* ***she*** *.*
- ***(8)*** *Even this relation in its simplicity is a* ***personification*** *of* ***things*** *and a* ***reification*** *of* ***persons*** *.*[38]
- ***(9)*** *It was not so* ***much*** *a matter of having power to* ***do*** *a* ***thing*** *as it was having the power to stop* ***things*** *from being done to you.*
- ***(10)*** *Baby,* ***you*** *'ve been so distant from* ***me*** *lately. And lately, don't even* ***want*** *to call* ***you*** *baby.*[39]

Examples 7 and 8 are two examples of salient semantic chiasmi it does not detect, whereas Examples 7 and 8 are two nonsalient "chiasmi" it extracted as candidates. This

[38] Karl Marx.
[39] Selena Gomez.

Word #1	Word #2	Cosine Similarity
he	she	0.870
me	want	0.757
much	things	0.752
victory	defeat	0.747
country	town	0.609
tree	computer	0.327
personification	persons	0.076

Figure 12: Pairs of words and their cosine similarity according to GloVe in its Common Crawl (48B) format, from highest to lowest.

problem is due to one simple cause: the embedding characteristics we used were too volatile, with similar words often having a relatively low cosine similarity (or opposite words having a low dissimilarity) while words with no clear links between them sometimes rated as particularly similar or dissimilar. This is shown in Figure 12, where we can see that similarity scores are only loosely tied to the actual closeness in meaning. We can also see that depending on the threshold, some candidates may or may not make the cute (for example, *victory* and *defeat* do not make the cut with our 0.75 threshold but would have with a 0.7 threshold). However, even a much lower threshold would have missed several candidates: it would need to be as low as 0.6 to detect Example 7, and no threshold would have been able to detect Example 8 due to its quasi-zero cosine similarity between *personification* and *persons*. This makes the current implementation of our tool still unfit in respect to salient chiasmi extraction, as such a task should have a recall close to 1 to be considered effective.

5.2 Deep Learning Models

This section will focus on the results of the different experiments described in Section 4.4. It will be separated in three parts: first, the results of GPTNeo 125M using our original dataset; second, the results of our first model on novel data; third, the results of a newly trained model using additional data gathered during the previous experiment; fourth and finally, the results of Bloom 560M using the same data as the third experiment for training and testing.

5.2.1 Original Experiment

As described above, our first experiment was to train, test and validate GPTNeo 125M on our original dataset - bar the antimetaboles from Dubremetz and Nivre (2016) -, using as negative examples random extracts from various corpora for a total of 638 positive examples and 1388 negative examples. Once entirely trained and tested, the model was then tested on 21 known positive examples, from the appendices of Dubremetz and Nivre (2016). As this test step only contained positive examples, no precision could be computed for this experiment (and thus, no F1 Score either).

The results of the model are visible in Figure 13. We can immediately notice that its results in-dataset are excessively high, beating all previous state-of-the-art but fall to a recall of only 81% with Dubremetz and Nivre (2016)'s antimetaboles (seventeen correctly annotated out of twenty-one) as soon as we move out of the dataset. As seen in the figure, this is still noticeably better than the previous state of the art regarding this

Model	Tested On	Precision	Recall	F1 Score
GPTNeo 125M	Regular Testing Split	98%	98%	98%
GPTNeo 125M	Dubremetz's Antimetaboles	-	81%	-
Dubremetz's	Dubremetz's Antimetaboles	-	69%	-

Figure 13: The results of GPTNeo 125M trained on our original dataset bar the antimetaboles from Dubremetz and Nivre (2016), compared to the results of Dubremetz and Nivre (2017).

Model	Tested On	Full Precision	Partial Precision	Partial Recall	Partial F1 Score
GPTNeo 125M	*Frankenstein*	2.0%	4.5%	50%	8.3%

Figure 14: The results of our fine-tuned GPTNeo 125M on the text of *Frankenstein*. The full precision is computed over the full 576 candidates, the partial metrics are computed over 150 fully annotated candidates.

dataset, Dubremetz and Nivre (2017)[40] but it is only one of the two important metrics for those tasks, as precisions is just as important as recall.

5.2.2 Using our Model on New Data

As described in Section 4.4.4, the next experiment we conducted was to use our newly trained GPTNeo 125M model on entirely novel data: the full text of *Frankenstein*, made of around 75,000 words or 3374 sentences. Of this full text, we extracted 576 sentences with at least one lexical criss-cross pattern and fed them to our model for classification. This experiment allowed us to better measure its performances - including its precision - on a numerous amount of data from a different genre.

Due to the high number of candidates, only the first 150 were manually annotated as salient, allowing us to compute a partial recall over those first 150 candidates. Moreover, all of the candidates classified as salient by the model were manually annotated as salient or not, giving us access to the full precision over the entire dataset of our model. Out of those, only two contained a chiastic pattern with some rhetorical saliency, which are shown below - although Example 12 was considered close to borderline by the annotators.

- ***(11)*** *I paused, examining and analysing all the minutiae of causation, as exemplified in the change from* ***life*** *to* ***death*** *, and* ***death*** *to* ***life*** *, until from the midst of this darkness a sudden light broke in upon me.*
- ***(12)*** *She* ***forgot*** *even* ***her*** *own regret in* ***her*** *endeavours to make us* ***forget*** *.*

[40] Schneider et al. (2021) obtained the same results as Dubremetz and Nivre (2017) regarding those antimetaboles.

The general results of this experiment are shown in Figure 14. As we can see, our model's precision on this new data is particularly low and its recall is only average - although with only two positive examples from which the partial recall can be computed, the results are not significant enough to draw any strong conclusion. Moreover, it is interesting to note that out of the two positive examples, it only detected the least interesting (Example 12) as salient while classifying the much more prototypical Example 11 as not salient.

5.2.3 Retraining a Model

The *Frankenstein* experiment brought to use a heap of new data in the form of 226 fully annotated examples[41] (3 positive, 223 negative) and 350 unknown examples that we considered as negative by default. To compare the results of our method when adding data from a new genre, we decided to train a new GPTNeo 125M model using exactly the same experimental setup as before - the only difference being the addition of those 576 new examples to the dataset. We'll call this new model GPTNeo 125M "*Frankenstein*" , whereas the original model will be called GPT 125M *Base*. The results of this experiment are visible in Figure 15.

As can be seen, those results are all worse than the original model's results, and the recall on Dubremetz and Nivre's antimetaboles is even lower than their 2017 approach. The most probable explanation is that the addition of new data to the training pool diluted the abilities of the model because of the wildly different literary style between *Frankenstein* and the other corpora, making it harder for the model to fall on any specificities of positive and negative examples both.

A follow-up experiment to this one was to compare the performances of this newly trained model to our original model on another novel's full text: "*Dracula*"[42] by Bram Stoker. The goal of this new experiment was to see if the added training data from *Frankenstein* would make the model perform better on a text from the same genre. In *Dracula*, the preprocessing tool detected 1426 potential antimetaboles - a particularly high number of candidates, even higher than in *Frankenstein*. Consequently, we chose to only annotate the candidates classified as salients, even though it meant only being able to compute the precision of the models for this specific experiment.

We can see in Figure 16 that, precision-wise, the "*Frankenstein*" version of GPTNeo 125M performed four times as well on the full text of *Dracula* as GPTNeo 125M *Base* did. This promising improvement is mainly explained by the much lower number of

[41] The first 150 examples, plus an additional 76 examples annotated as false after being labelled as salient by our model.

[42] Around 160,000 words, or 9800 sentences.

Model	Tested On	Precision	Recall	F1 Score
GPTNeo 125M *"Frankenstein"*	Regular Testing Split	95.4%	91.2%	93.3%
GPTNeo 125M *"Frankenstein"*	Dubremetz's Antimetaboles	-%	57.1%	-%

Figure 15: The results of GPTNeo 125M *"Frankenstein"* on both its regular test split and Dubremetz and Nivre's antimetaboles.

Model	Tested On	#SalientAnnotation	#NonsalientAnnotation	Precision
GPTNeo 125M *"Frankenstein"*	*Dracula*	63	1363	12.7%
GPTNeo 125M *Base*	*Dracula*	308	1118	3.6%

Figure 16: The results of GPTNeo 125M "*Frankenstein*" on the full text of *Dracula*. "#SalientAnnotation" (resp. "#NonsalientAnnotation") is the total number of candidates categorised as salient (resp. nonsalient) antimetaboles by the model.

miscategorised nonsalient antimetaboles: while our original model categorised as positive 308 antimetaboles, the *"Frankenstein"* model only found 63. An interesting development however is that our original model detected 3 more true positives, adding up to a total of 11 salient antimetaboles correctly classified (against 8 correctly classifier by the GPTNeo 125M "*Frankenstein*"). This information allows us to compute an upper bound of ***72.3%*** for the recall of our newly trained model, since we know that at least 11 candidates were salient antimetaboles.

5.2.4 Comparing Two Transformers

The last experiment we conducted was to train an entirely new architecture of model, Bloom 560M, on the same data and to test it on the same corpora as GPTNeo 125M "*Frankenstein*" was. Since the training process was exactly the same as GPTNeo 125M "*Frankenstein*" , we decided to call this new model Bloom 560M "*Frankenstein*" . In practice, the goal of this experiment was to compare the performances of those models, and ensure that the low out-of-dataset results we obtained on the previous experiments were not due to specificities of their architecture.

Once the model was fine-tuned, it was tested against the same testing set as before: the regular 10% testing split, Dubremetz and Nivre's antimetaboles and the 1426 candidates from *Dracula*. The results, visible in Figures 17 and 18, show that Bloom 560M "*Frankenstein*" performed worse than GPTNeo 125M "*Frankenstein*" in all trials but Dubremetz and Nivre's antimetaboles[43]. However, in that experiment, it still performed

[43]An amusing detail, however, is that it serendipitously classified as antimetaboles two examples - one

Model	Tested On	Precision	Recall	F1 Score
Bloom 560M *"Frankenstein"*	Regular Testing Split	87.3%	98.2%	92.4%
Bloom 560M *"Frankenstein"*	Dubremetz's Antimetaboles	-%	66.7%	-%

Figure 17: The results of Bloom 560M *"Frankenstein"* on both its regular test split and Dubremetz and Nivre's antimetaboles.

Model	Tested On	#SalientAnnotation	#NonsalientAnnotation	Precision
Bloom 560M *"Frankenstein"*	*Dracula*	137	1289	5.8%

Figure 18: The results of Bloom 560M "*Frankenstein*" on the full text of *Dracula*.

worse than GPTNeo 125M *Base* and Dubremetz and Nivre (2017)'s model. An important detail however is that it still correctly classified as much salient antimetaboles as GPTNeo 125M "*Frankenstein*" on *Dracula* (8, out of at least 11), but its precision is twice as low because it also outputted double the amount of false positives.

With Bloom 560M being worse on almost all fronts than GPTNeo 125M, the general trend of transformers models having trouble to properly perform the detection of antimetaboles is reinforced. This could be caused by several different reasons, that we will explore in the next chapter.

5.3 Compiling the new data

Before discussing the results of our previous experiments, there is one additional - and welcome - outcome that resulted from them: a heap of new data. This data is separated as follow:

- 15 antimetaboles, with 4 from *Frankenstein* and 11 from *Dracula*.
- 1 semantic chiasmus, from *Dracula*.
- 1 phonetic chiasmus, from the "joke" corpus (Pungas, 2017).
- 1853 negative or unknown (and thus considered as negative) examples, from *Frankenstein* and *Dracula*.

All of the new data from *Frankenstein* and *Dracula* were saved in a subfolder of our aforementioned chiasmi dataset repository[44], and the phonetic chiasmus was updated as

sentence from *Dracula* and one from the joke corpus - that actually are a semantic chiasmus ("She wants blood, and blood she must have.") and a phonetic chiasmus ("What do you call two nuts on a chest? Chestnuts.").

[44]Direct link to this subfolder: `https://github.com/Dironiil/ChiasmusDatasets/tree/main/data/new`

such (from "NotAChiasmus") in the original dataset files.

This new data thus brings our final corpus of chiasmi to a total number of 780 salient chiasmi (subdived into 664 antimetaboles, 99 semantic chiasmi and 9 phonetic chiasmi), 4675 random extracts with some lexical criss-cross pattern of which 1955 are sentences from *Frankenstein* and *Dracula* and 1388 sentences or pair of sentences which were not analyzed for any chiastic structure.

6 Discussion: limitations and future work

This chapter's goal is to discuss the limitations of our work, what could be done to improve upon it and what new areas could be researched. It will be subdivided in three main sections, with the first one talking about the candidate extraction process, the second one about the various deep learning experiments we ran and the last one focusing on possible future works.

6.1 Discussing our Candidate Extraction Tool

As presented in the previous chapter, our candidate extraction tool showed mixed results: while it detected most of the antimetaboles, its embedding part fell entirely flat at properly extracting semantic chiasmi. Moreover, its current implementation is relatively slow and could be optimised, especially to work on very large input like Dubremetz and Nivre did on the *Europarl* corpus.

6.1.1 The hard Task of Detecting Semantic Chiasmi

6.1.1.1 What Went Wrong

Our tool did not manage to properly detect semantic chiasmi for one simple reason: the measure we used to do so was too volatile and imprecise. The previous chapter showed that our embedding model, GloVe (48B, common crawl), gave cosine similarity between words only loosely tied to their actual distance in meaning. Although the actual reasons for this are unknown, two can be hypothesised.

First, embedding models are trained using contextual data. The vector representations of words are learned from one of two techniques: either *Continuous Bag Of Words* ("CBOW"), in which the model has to guess a word depending on the words around it or *Skipgram*, in which the model has to guess the words around a given one. This means that words which have a high chance of standing in the same sentences have a higher probability of having a common embedding (like *victory* and *defeat*) while words which will very rarely be seen in the same contexts will have a low or even negative cosine similarity (like "personification" and "persons") - even though the first pair of words are literal antonyms whereas the second pair stems from the same abstract concept. Although cosine similarity is often presented as a way to find words with similar or dissimilar meaning, it can frequently instead tells us if the words are *similarly* used - a correlated but not entirely equal measure, as words can be used in the same context while being wildly different in meaning and vice versa.

Second, the automatic translation of words into a mathematical embedding of their meanings is a very hard task. Words often have different meanings depending on their context, but the embedding model we used could only assign one embedding vector for each word. Some words can even be contronyms (or auto-antonyms), that is words with opposite definitions: an excellent example of this is the verb "to sanction", which can both mean "to grant approval" and "to condemn". On those, *GloVe* is simply unable to compute a proper embedding, and any softwares following up on its results would receive improper information.

Due to the second point in particular, the choice of the embedding model is highly important for each task. In our case, we opted for *GloVe* with a 48B embedding in its common crawl version (i.e. trained on a wide range of data) because our task had to deal with words from very varied contexts. As such, we didn't want embedding models trained on a specific genre of data that could miss the meaning of words in contexts they did not know. However, this does not mean that this version of GloVe was the best embedding software for our task, as we will see in the next section.

6.1.1.2 How Our Pipeline could be Improved

As mentioned in the previous paragraph, which model to use is an important decision for any NLP tasks involving embedding. We can not be certain that the one we chose, GloVe 48B Common Crawl, was superior to regular embedding models (such as *Word2Vec* or *FastText*, two other ML models for word embedding) on our specific task. Implementing versions our tool with other embedding models could thus be the first way to try to improve our pipeline's performances.
Furthermore, there now exists newer embedding techniques that take into account the context of words and encode it into their embeddings, such as *CoVe* or *RoBERTa*. It is probable that those models could perform better for our specific task: we were indeed searching not for words with a general similarity between them, but for words which are similar given a specific context. We did not know enough about them at the time to choose them over the more known and recognised *GloVe*, but we encourage future researches to look into them for embedding in regard to semantic chiasmi.

An other possible way to improve the extraction of semantic chiasmi candidates is to use an entire method entirely. We thought about two in particular, but were not able to implement them: the first one is a system based on a synonym / antonym dictionary; the second one is based on etymological roots of words. Those two approaches are described with greater details below:

- Similar to our lemmata approach considering as antimetabole candidates any text with an inverse repetition of equal lemmata, the synonym / antonym method

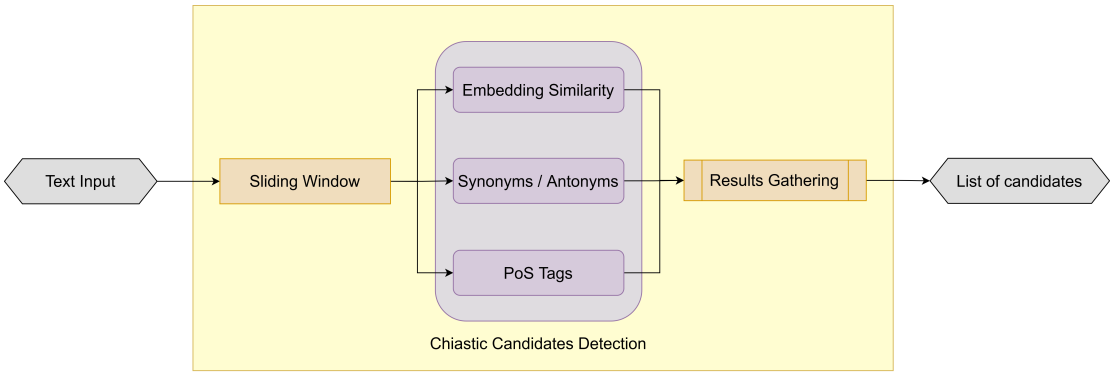

Figure 19: A proposal for an enhanced extraction system of semantic chiasmi candidates parallelising several methods. The result gathering step can either be a consensus system, or a simpler "logical or" forwarding any positive results indiscriminately.

would consider as semantic chiasmi candidates any text with an inverse repetition of synonyms and/or antonyms. Much like Schneider et al. (2021)'s approach to detect possible chiasmi, this process would not have a perfect recall as a lot of salient chiasmi are not made of perfect synonyms or antonyms. However, it would still retrieve a high number of interesting candidates while yielding a much lower number of entirely uninteresting one. This approach would, for example, retrieve a chiasmus such as *"He **arrived** in **victory**, and in **defeat departed**"*.

- The second method would look at the etymological roots of words to retrieve a possible semantic chiasmi. This approach aims to detect chiasmi similar to Example 8 where "persons" and "personification" are words with the same root, while having different meaning and sometimes PoS tags. Although it is hard to gauge how many salient chiasmi could be retrieved with this method, it could be a welcome addition to other approaches nonetheless to cover for some of their weaknesses.

Finally, another interesting way to improve the extraction of semantic chiasmi could be to mix several different techniques. For example, Schneider et al. (2021)'s idea of using PoS tags is inherently incomplete as some chiasmi are simply not composed of an inversion of PoS tags but it can definitely detect a high number of them nonetheless - at the price of an excessively high number of true negatives that can clutter classification or ranking pipelines. As long as they are robust against a high number of nonsalient examples however, the PoS tags approach method could then be used *in tandem* and not *instead* of the improved embedding techniques we described just above, with each completing some of the other's blind spots. In particular, if enough different methods are used in parallel, a consensus system (or even a weighed consensus) can be imagined, where a candidate is considered interesting enough to be extracted only if enough dif-

ferent techniques detected it. Such a multi-approach system can be visualised in Figure 19.

6.1.2 Optimising the process

Our main extraction tool was relatively slow, detected candidates that could have been ignored and its output was not usable for all kind of ML training. However, all those problems could have been improved upon given more time and work on it:

- Its slow runtime was due to some poor choices of implementation, mainly in its main processing loop and in the way the files were loaded and parsed for lemmata and other tags. While the main processing loop would be quite hard to entirely refactor and optimize, the loading and parsing part could have been. In particular, in the current version of the tool, a file is entirely loaded and parsed before the candidate detection process can begin. This process is inefficient, and files could instead have been loaded and parsed on the go, saving a lot of memory and consequently time for the biggest files like very large corpora. Indeed, whereas Dubremetz and Nivre (2017) were able to process a large part of the *Europarl* corpus, our tool had troubles parsing 7.5MB files on a computer built in 2021.
- Moreover, the way the detection was implemented was not perfect: mainly, our tool looked for candidates in any 30-words windows, without any sequence of characters breaking this search. This is particularly problematic when processing files with a lot of small, independent extracts, as possible candidates over two (or sometimes more) extracts could be detected even though they obviously can not be salient chiasmi. This could have been improved through a medium refactor of our tool: by asking for a list of "break" characters after which the sliding window should be reinitialized, this problem would almost entirely vanish. For example, if a file is made of extracts separated by a blank line, then the double character sequence `\n\n` could have been given as a break sequence to keep the window from sliding continuously over several examples.
- Finally, the easiest to solve out of those three problems is the output of our tool. In its current version, its output is composed of: the raw text of the candidate extract with some extra characters on its left and right for context, the character position of the members of the chiasmi candidate, their index in the sentence, the position of the candidate extract in the greater file, their pos tags and finally two lists, one being all the words in the candidate, and one being all their lemmata.[45] While

[45] As this thesis could not make use of this output, the datasets we presented earlier are not parsed according to it. More details about parsed data can be found in Meyer (2023), which focused on classical ML methods and thus used our tool's output directly.

this output presents a lot of information usable by different statistical approaches, one is sorely lacking: the entire sentences over which the candidates exist. This proved a problem twice: once during the annotation process, since it is harder to annotate candidates as salient or not without their full sentences for context and afterwards during the training process, since the transformer models trained in this thesis wanted inputs made of full sentence, we could thus not use our main tool as a way to extract more negative examples from random texts. By adding this information, our tool could be made more general and usable for a larger variety of tasks.

In general, an entire refactor of our tool using better techniques could yield a largely improved software and thus the entire detection, annotation and training pipelines imagined in previous chapters.

6.2 Finding more Salient Chiasmi, more Easily

In Figure 9, we presented a possible pipeline to collect more humanly annotated data, more easily. However, we were never able to properly run it with the current version of our tool and the models we had at our disposal at the time. Despite those problems, this idea is still in our opinion an excellent way to improve the amount of data regarding chiasmi (or in general, regarding other rhetorical figures with different candidate extraction softwares and classification / ranking tools).

For example, such a pipeline could be implemented using the model from Meyer (2023). His method of detection, building upon the existing researches on classical ML, shows promising results and could thus be used as a base to annotate more antimetaboles.

However, it is highly important to keep in mind that this proposed method to gather more data is not foolproof: it may construct datasets systematically biased against some chiasmi. Indeed, by using the existing models to detect new examples of chiasmi, any bias those models might have against rarer forms of chiasmi (for example, chiasmi with an unusual syntactic pattern) will be repeated in the resulting dataset. If such a bias ends in a dataset, then any models trained on it would reproduce it and any further datasets produced while using them would repeat it, creating a vicious circle.
As such, the research for more examples of salient chiasmi should not rely entirely on this proposed semi-automated method. Moreover, when some biases are discovered with a given model, they should be openly presented and discussed so that following works on the subject could take them into account.

6.3 Transformers for Antimetabole Detection: a Good Idea?

As we have seen in the previous chapters, the transformers we trained produced very poor results as soon as they left their original datasets. This section will try to understand how and why this happened, how it could be corrected and, in general, how future works could approach the use of ANN and DNN for the detection of antimetaboles or even chiasmi at large.

6.3.1 The Failures of Transformers on Antimetabole Detection

As we have previously described, transformers have taken upon a large part of NLP over the last four years. This was possible thanks to their incredible results on a high number of tasks, often greatly improving upon the previous state of the art. Our failure to make them on the task at hand is thus even more noticeable, but can actually be explained by several points. Those points will be explored from the most general, about transformers at large, to the most specific, about our experiment in particular.

The first possible cause of our poor results is the inherent way transformers work. Whereas the classical ML approaches from Dubremetz and Nivre (2017) and Schneider et al. (2021) used for input specific features extracted from the candidates, transformers only ever take as inputs "raw texts" (or rather, their tokenized version). This means that no task-specific features can be used to enhance the data on regular pretrained transformers: they can only look at lists of tokens and try to learn the necessary properties out of those list. This works well for a lot of NLP tasks, but not all can be deduced entirely through general learning. Moreover, transformers' attentions are, by default, equally spent on each words and punctuation symbols for a given input. This means that any tasks focusing on specific words of a given input would be harder to solve for a transformer.

The second possible cause (and probably the most impactful) is another inherent property of transformers that we did not mitigate properly. Indeed, transformers are permutation invariant, i.e. not sensitive to the order of the tokens in their input. The two inputs *"I am you"* and *"am I you"* would for example be considered the same way by any regular transformer. This comes from the fact that transformers process text inputs not by reading them words per words, but by looking at each words in relations with every other words in the input - both forward and backward. When this permutation invariance becomes an issue, the positions of the words from the raw inputs are usually encoded either in the input during the tokenisation process, or using specific attention matrices within the transformers themselves (Chen et al., 2021).

Input Text	Corresponding Embedding
It is a squirrel.	`[ 632, 318, 257, 33039, 13 ]`
squirrel is. It a	`[ 33039, 318, 13, 632, 257 ]`

Figure 20: The results of the GPTNeo Tokenizer when used on text inputs with shuffled word.

Something we noticed at the very end of our thesis, however, is that both the tokenizers we used both for GPTNeo 125M and for Bloom 560M do not encode those position in the input - and neither were there in our knowledge any mechanisms to transfer such an information to the attention modules within the transformers. This can be seen in Figure 20, where two inputs containing the same word (one sentence and a shuffle of it) have the same tokens for each given word - like "632" for the word "It" (first and fourth position, respectively). This if of course a large oversight, as chiasmi in general and thus antimetaboles in particular are highly sensitive to the order of words: while "death to life, life to death"[46] is a proper and even prototypical antimetabole, "death to death, life to life" is not a salient antimetabole, or even a chiasmus at all.

Finally, the third possible cause for our low results could the way with which we handled our training data: while we had a high number of both positive and negative examples, most of those did not came from the same sources. What this implies is that positive examples might have more key differences with negative examples than simply being salient antimetaboles. As such, our transformers could possibly pick as relevant features for classification details in the text that are not actually relevant to being an antimetabole, like the vocabulary, the length of the input or specific punctuation. This last point was directly noticed during some manual experiments, as the almost identical inputs "**East** is **West**, and **West** is **East**" and "**East** is **West**, and **West** is **East**." were classified differently by GPTNeo 125M *base*, with the non-punctuated one being incorrectly classified as nonsalient.

Using such out-of-task features would explain why they appeared so effective when tested within their dataset, but proved less effective when tested on similar but out of dataset antimetaboles (those of Dubremetz and Nivre (2017)) and entirely ineffective when testing entirely out of genre, on the two novels. It would also explain why the addition of the data from Frankenstein to the training set made GPTNeo 125M "*Frankenstein*" worse on almost all front compared to GPTNeo 125M *Base* : as this data included both negative and positive examples from the same source, there was not as much out-of-task features that the model could train on and it was thus left with a harder (although closer to the actual) task than usual.

[46] From *Dracula*.

In general, this problem of data could also explain - in part at least - why our models had such a low precision when it came to the novels (as a reminder, the best precision we obtained on this task was 12.7%, by GPTNeo 125M "*Frankenstein*" on *Dracula*). Most of their training data were not from literary pieces, but examples with salient antimetaboles tend to show a higher "rhetorical interest" than the examples without - a rhetorical interest that is also found in quantity within both novels, with their authors being highly renowned for their works. If their training made them look for features more indicative of rhetorical interest than of the actual presence of antimetaboles, the models would hence be misled on a text where most sentence are carefully crafted.

This problem of training on the wrong features can also be envisioned within a perspective entirely oriented on rhetorical figures. As Harris et al. (2018) presents, what we call salient chiasmi are never alone. They are always accompanied with several other figures that usually enhance their effects - when those additional figures are not inherent to the "main figure", like the repetition of lemmata within antimetaboles. Some of the most common co-figures are, for example, parisons[47], mesodiploses[48] or anaphoras [49]. A model trained on antimetaboles could thus be misled and "think" that it is looking for repetitions or parisons instead, if most of its positive examples show them while most of its negative examples do not. This is something that we have actually noticed, as a substantial number of false positives from the novels presented some form of repetitions, or at least sentence structures that could be those of an antimetabole.

6.3.2 Possible Improvements to our Work

The first and most obvious way to improve the performance of transformers for antimetabole detection would be to correct the problems listed in the previous section. Out of those three, the first one would be the hardest to solve as it is the most intrinsic to transformers. Transformers are inherently made to look at sequences of words, and using additional features would either need for them to be encoded within the token sequences fed to the transformers, or to create a new architecture for a transformer - and therefore train it from scratch, a very data and computing power hungry process.
The second one, however, could be simply solved by a better implementation of our tools: using the latest and most effective position encoding techniques, such as the one described in Chen et al. (2021), could highly improve the results of similar models to ours on the same task.
Finally, the third problem would be harder to entirely correct, but could be mitigated. To entirely correct the bias in the data, negative (resp. positive) examples should be

[47] A parison happens when a given syntactic structure is repeated over a serie of clauses or sentences.
[48] A mesodiplosis is the repetition of one or several words in the middle of several clauses or sentences.
[49] An anaphora is the repetition of words at the beginning of several clauses or sentences.

taken out of all sources that were used to find positive (resp. negative) examples. This is an almost impossible task, seeing as a lot of our examples are anonymous and their original source unknown. Another solution could thus be to simply cut out all the data which do not have oppositely labeled examples from the same source but this would be impractical, as the less there is training data, the less effective the resulting models for almost all ML tasks. However, this problem can be mitigated in two ways: first, by expanding the number and genres of sources from which we draw our positive and negative examples, and by expanding the scope of negative examples. What we mean by this second point is that negative examples containing figures often associated with antimetaboles should be added to the training data. The first half would mitigate the possibility for models to overfit on specific features of the examples, as there would be much more variety within them, while the second would keep them from training on co-figures instead of antimetaboles themselves, as those co-figures would also be present among the negative examples without the actual point of focus.

Another way that could improve the performances of transformers on our task would be to take on the idea of Dubremetz and Nivre (2015) and use a ranking system instead of a classification system. In their article, they quite adequately described how the question of saliency within rhetorical figures is not black or white but rather a graduated phenomenon, and that any figure should not be treated as either salient or nonsalient, but only as more or less salient than another figure. With this argument in mind, they decided to create a classical ML model that would not classify the antimetaboles in a given input, but only graduate them from most salient to least salient. Using such an approach with transformers would allow for finer results, ranking the antimetaboles whose salience is most confidently assessed by the models higher - and thus more easily accessible. Moreover, the use of a ranking system would also allow for more and better metrics to be used, such as the average precision[50] of a model, or precision / recall graphs. A similar approach would be to add a "certainty" output to our classifier model, indicating the certainty with which the model classify a certain object in a given category.

As an additional point, the choice of training hyperparameters can also impact the final performances of any ANN. Parameters like the number of epochs, batch size, learning rate or the size of the model can make or break a given experiment. As we have not used different hyper parameters on the same models for our experiments, we cannot know how much our choice impacted their results. This is an issue that could be more

[50]The average precision is the mean of precisions at several threshold within the ranked information, weighed by the differences in recall. It was first proposed for information retrieval but works well for any task where a ranking model can gradually recover more positive examples the further down the ranks we look.

systematically explored in the future.

Finally, it is important to remember that while transformers have revolutionized many researches within the field of NLP, they are not the correct solution for every tasks. In this specific case, the task at hand might simply be too specific for generalist models like them to work on it directly. Their results may (and will probably) improve on this specific task, but keeping in mind that other architectures exist is crucial.

6.3.3 ANN and DNN for Antimetabole Detection

As said in the previous paragraph, transformers are not "the end of history for Natural Language Processing" (a quote from the title of Chernyavskiy et al., 2021): despite their good results in NLP, other neural networks techniques could be used for the task at hand. This idea can be divided in two halves:

- On one hand, simpler ANN or DNN could be used with the features studied by Dubremetz and Nivre (2017), Schneider et al. (2021) and Meyer (2023). On several tasks, well-crafted ANN are the state of the art instead of classical ML methods or large and generalist DNN. Due to its specificities, the detection of salient antimetabole might be one of them.
- On the other, more complex DNN looking at texts as a whole like CNN or LSTM networks might have a better grasp on this specific task than transformers due to inherent properties. Each of these two specific examples in particular have reasons to be studied: CNN are made to look for specific patterns, independent of the size or position of these patterns within the complete input; and LSTM are recurrent networks particularly good at processing ordered series of inputs while focusing on specific relations withing those series.

In general, the failure of transformers to properly fulfil the task at hand in this thesis should not discourage future researches to look deeper into ANN and DNN as possible solutions for the detection of antimetaboles - or chiasmi at large. Rather, the problems encountered here can be used as foundations for the development of more robust, effective and efficient systems.

6.4 A few Additional Ideas for Future Works

This section will develop a few additional proposals for future works regarding the detection of antimetaboles. The first one is based on the expansion made by Schneider et al. (2021), the second stemmed from a discussion with Pr. Randy A. Harris, from the University of Waterloo, the third came from a discussion with Yohan Meyer in which

we talked about the respective results of our approaches and the fourth from some early ideas we finally decided not to work upon during this thesis.

6.4.1 Expanding the Search to Semantic Chiasmi

Salient antimetaboles are only one small part of a larger family and, although they are the easiest to detect and the most common among chiasmi, their detection is only one step in larger walk towards a full understanding of chiasmi. Exploring chiasmi at large was already tried in the past by Schneider et al. (2021), although they are - to the best of our knowledge - the only authors to date who have tried such a task. However, we have not been able to immediately follow in their footsteps due to both a lack of data and an improper candidate extraction tool.

Researching the effectiveness of transformers or other DNN techniques regarding the detection of salient semantic chiasmi could however yield highly interesting results. As transformers are inherently looking at the words through semantic and relational lenses, models which are trained on semantic chiasmi while taking into account the possible improvements we've listed above have a chance to surpass the current state of the art.

6.4.2 Understanding the What and Why of Rhetorical figures

The second proposal originates from a point already described in Section 3.3 that originated from Pr. Harris and Jelena Mitrović: according to them, complete researches on any topics within the field of Computer Linguistics should not only focus on the forms of the objects but also their functions. In our case, salient antimetaboles are not salient for no higher reasons, they are salient either because they convey a new meaning that would be absent were it not for their presence, because they enhance an already present meaning or because they make their host piece more pleasing to the ear - often even, because of several or all of those reasons. In Example 13, the antimetabole is vital to the joke, as it shows the opposition between "the optimist" and "the pessimist" views while also adding to the humor because of the close structure between both clauses despite their opposite meaning.

- ***(13)*** *An optimist goes to the window every morning and says, 'Good* ***morning*** *,* ***God*** *!' The pessimist goes to the window every morning and says, 'Good* ***God*** *!* ***Morning*** *!*

- ***(14)*** *For years* ***I*** *have loved* ***her****. For years* ***she*** *has loved* ***me****.*[51]

Following Pr. Harris arguments, one way the current state of the art on antimetabole detection could be improved is to work on tools that would not (or not only) detect salient antimetaboles within a text but also the reason they are salient in the first place. As Examples 13 and 14 show, antimetaboles can fulfill opposite roles within a rhetorical unit: while the first one shows an opposition between the two points of view, the second shows how both characters reciprocate each other's feelings. Having the knowledge on the why an antimetabole is present in the first place (or any other rhetorical figure) is thus highly important for a lot of higher level tasks, like sentiment analysis or semantic relationing, while also being useful for any human users of such a tool.

6.4.3 Combining Two Approaches

The final proposal comes mostly from the intertwining of Yohan Meyer's work and my own: as we've closely worked together for several months and still stayed in contact after entirely splitting the subject of our theses, we were able to compare the results of our theses. From those comparisons stemmed an idea: transformers and classical ML models look at the problems from two very different perspectives. As such, they could possibly be combined to fill each other's blindspots: the lack of specific features and attention towards specific words and orders with transformers, and the lack of consideration for the general sentence(s) at hand with classical ML methods.

Three possible mixed architecture can be envisioned:

- The most simple of the three is to put them in a sequential pipeline, where the input is first fed to one model and then to the other if the first one deemed it salient. A good example would be to feed the input to a transformer generally trained to detect "antimetabole-like" sentence structures, before feeding the most likely candidates to a classical ML model. This approach would improve the precision by filtering more true negatives, but could come at the cost of recall as positive candidates flagged as "negative" by any models would be discarded.
- The second approach is to feed candidates to both models at the same time, and to consider as salient any candidates flagged positive (or above a certain rank, in the case of ranking models) by one model *or* the other. Opposite to the first one, this architecture would improve the recall, by allowing candidates to appear even if they were ignored by one of the approach, at the cost of precision.
- The last and possibly most interesting architecture would be to use one model's output as the input for the other. The easiest scenario to imagine would be a

[51] Arthur Conan Doyle.

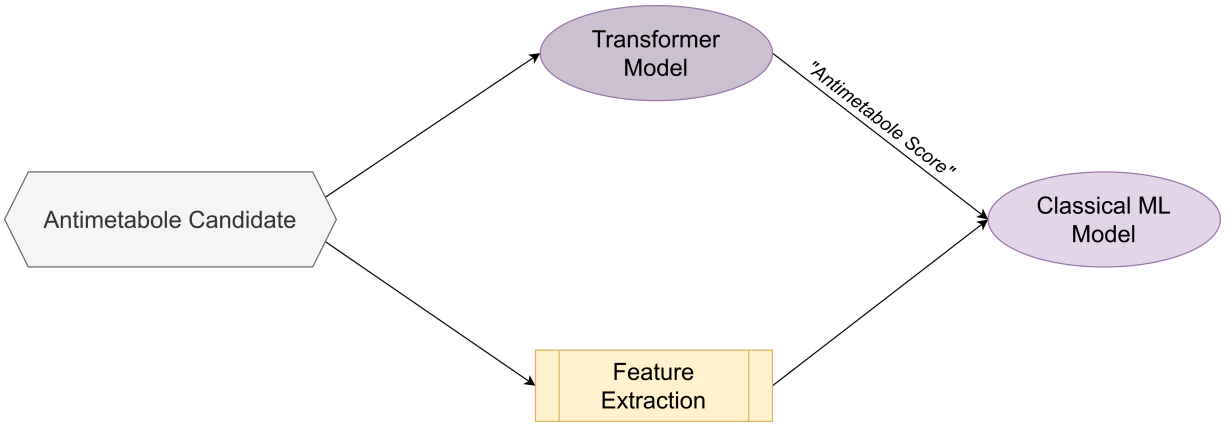

Figure 21: A possible architecture to combine both classical ML and transformers: a transformer providing additional features to a classical ML model.

pipeline in which a candidate is, at the same time, both transformed into features and fed into a transformer trained to score the candidate on a given chiasmi-related task. The candidate's features and the score are then all fed to the classical ML model as input features. This would allow the classical ML model to take into account the transformer's score on the candidate, a score computed with the whole sentence in mind and which may thus show more than what the base features can.

In particular, a visualisation of the last architecture is visible in Figure 21. Unlike the first two approaches described above, combining both models together could theoretically improve both precision and recall. This makes it the most promising out of the three, especially as a well-crafted pipeline could draw from the state of the art for the regular features while improving upon it thanks to the transformer model.

6.4.4 Using Transformers to Augment the Available Data

At the very beginning of this thesis's research, one of the possible path was not to work on *detecting* salient antimetaboles within any given texts, but to specifically look for way to *augment* the existing data on antimetaboles. Although we finally did not explore this idea directly, it is still a promising way to improve our effectiveness regarding salient antimetabole detections as most ML models work better the more training data they have.

Transformers, in particular, would be particularly adapted to such a task as they were first designed to be used in sequence-to-sequence tasks. By training a transformers on a high number of positive and negative examples of salient antimetaboles, it could generate more examples which could then be manually or semi-automatically annotated. Whether those examples end up being positive or negative, they would still be

a good addition to the existing data: positive examples would give more antimetaboles on which to train other models, whereas negative examples would force other Deep Learning models to "understand" where the generating transformers failed to create an antimetabole.

7 Conclusion

In this thesis, we explored how Deep and Transfer Learning could be used to detect salient antimetaboles within English texts. More specifically, our main research focused on the effectiveness of large and generalist pretrained transformer models such as GPTNeo and Bloom when fine-tuned on salient antimetabole classification with a relatively large dataset (around 2,000 examples). In order to do that, we created a modular Python 3 notebook pipeline to load all necessary data from distant repositories, fine-tuned our models using publicly distributed, generalist, pretrained transformers (respectively GPTNeo 125M and Bloom 560M) as bases, test them in-dataset, out-of-dataset on antimetaboles from other works and entirely out-of-genre on two novels in the public domain (*Frankenstein*, by Mary Shelley and *Dracula*, by Bram Stoker). To implement and train the transformers, and allow for the high modularity of our pipeline, we used Huggingface's *transformers* library (Wolf et al., 2020b).

Our results on the main question were mixed at best: as we have shown, transformers can quickly overfit on specificities within the training data, showing excellent results in their original datasets but mediocre or even bad ones when tested outside of them. In particular, their precision on out-of-genre data was particularly low, due to the very high number of sentences they considered as salient antimetaboles because of, we supposed, other interesting but not directly related rhetorical features within them. However, we concluded that those low results are not necessarily cause not to dig further in this direction, as several parts of our work could be improved upon or entirely changed to yield better results.

However, our research process also yielded two notable side-results: a candidate extraction - detection - annotation semi-automated pipeline that could be used for future researches, and a very large dataset of various chiasmi including almost 800 positive examples of chiasmi, among which can be found slightly under 700 antimetaboles. In particular, this dataset is an immense improvement from the previous best available collection of data in English, the 31 antimetaboles from Dubremetz and Nivre (2017) and can thus be used for a very wide array of Machine Learning tasks, from classical Machine Learning to a variety of Deep Learning techniques. Our proposed extraction - detection - annotation pipeline can moreover allow for even more data to be collected, more easily - further easing the sore problem that was the lack of data regarding chiasmi.

From this point onward, future works basing themselves upon this thesis could thus take one of several paths, with a few of those described here. The first one would be to use the same pipeline as we did, but improve upon the several shortfalls we have noticed to see the results of those improvements on the actual results of the models.

The second would be to extend our method to semantic chiasmi in general and compare those results to Schneider et al. (2021), using the non-negligible 100 semantic chiasmi we have collected as training data. The third one would be to look into other Deep Learning methods for antimetabole (or even chiasmi) classification and compare them both to transformers and classical ML methods. The fourth one would be to look how to join classical ML methods and DL / TL methods into one single pipeline to take on the qualities of both while mitigating each other's weaknesses.

To conclude, we have shown in this thesis that transformers are not an easy and perfect fit regarding the classification of salient antimetaboles. However, the field should nonetheless be explored deeper as the proper detection, classification and eventually explanation of rhetorical figures is a particularly important process for several NLP high-level tasks such as but not limited to sentiment analysis, literary essays grading or writing assistants. The past fifteen years have shown how the research community can obtain excellent results on tasks that were entirely unresearched before, and we hope that this thesis have provided, for anyone who wish to further our understanding of antimetaboles and chiasmi, a fundation and a glimpse into the steep progress of NLP over the last decades[52].

[52] One could recognise here a chiasmus that no research to date could detect: the last words of this thesis, "the steep progress of NLP" mirror the very first words of its introduction: "NLP has progressed steeply".

References

ALPAC. (1966). Language and machines: Computers in translation and linguistics. *National Academy of Sciences, National Research Council*.

Benello, J., Mackie, A. W., & Anderson, J. A. (1989). Syntactic category disambiguation with neural networks. *Computer Speech & Language*, *3*(3), 203-217. Retrieved from https://www.sciencedirect.com/science/article/pii/0885230889900181 doi: https://doi.org/10.1016/0885-2308(89)90018-1

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A neural probabilistic language model. In T. Leen, T. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems* (Vol. 13). MIT Press. Retrieved from https://proceedings.neurips.cc/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf

Bengio, Y., Frasconi, P., & Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. In *Ieee international conference on neural networks* (pp. 1183–1188).

Berger, A., Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Gillett, J. R., Lafferty, J., ... Ures, L. (1994, March 8-11). The candide system for machine translation. In *Human language technology: Proceedings of a workshop.* Plainsboro, New Jersey.

Black, S., Leo, G., Wang, P., Leahy, C., & Biderman, S. (2021, March). *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.5297715 doi: 10.5281/zenodo.5297715

Chen, P.-C., Tsai, H., Bhojanapalli, S., Chung, H. W., Chang, Y.-W., & Ferng, C.-S. (2021, November). A simple and effective positional encoding for transformers. In *Proceedings of the 2021 conference on empirical methods in natural language processing* (pp. 2974–2988). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.emnlp-main.236 doi: 10.18653/v1/2021.emnlp-main.236

Chernyavskiy, A., Ilvovsky, D., & Nakov, P. (2021). Transformers: "the end of history" for natural language processing? In N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, & J. A. Lozano (Eds.), *Machine learning and knowledge discovery in databases. research track* (pp. 677–693). Cham: Springer International Publishing. doi: 10.1007/978-3-030-86523-8_41

Chomsky, N. (1957). *Syntactic structures*.

Chomsky, N. (1965). *Aspects of the theory of syntax*.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language process-

ing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(ARTICLE), 2493–2537.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Davies, M. (2008, updated March 2020). *Corpus of contemporary american english.* Retrieved 2022-12, from `https://www.english-corpora.org/coca/`

Dechter, R. (1986). Learning while searching in constraint-satisfaction-problems. In *Proceedings of the fifth aaai national conference on artificial intelligence* (pp. 178–183).

Dubremetz, M. (2013). Towards an automatic identification of chiasmus of words. In *Proceedings of recital* (pp. 150–163). (Article in French)

Dubremetz, M., & Nivre, J. (2015, 01). Rhetorical figure detection: the case of chiasmus. In (p. 23-31). doi: 10.3115/v1/W15-0703

Dubremetz, M., & Nivre, J. (2016, June). Syntax matters for rhetorical structure: The case of chiasmus. In *Proceedings of the fifth workshop on computational linguistics for literature* (pp. 47–53). San Diego, California, USA: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/W16-0206` doi: 10.18653/v1/W16-0206

Dubremetz, M., & Nivre, J. (2017, May). Machine learning for rhetorical figure detection: More chiasmus with less annotation. In *Proceedings of the 21st nordic conference on computational linguistics* (pp. 37–45). Gothenburg, Sweden: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/W17-0205`

Epstein, R., Roberts, G., & Beber, G. (Eds.). (2009). *Parsing the turing test*. Springer.

Fass, D., Lesgold, A., & Patel, V. (1997). *Processing metonymy and metaphor*. Ablex Publishing Corporation Greenwich, CO.

Friedman, L. (2020). *arXiv STEM scholarly articles.* data.world. Retrieved 2023-01, from `https://data.world/liz-friedman/arxiv-stem-scholarly-articles`

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267–285). Springer.

Gawryjolek, J. (2009). *Automated annotation and visualization of rhetorical figures* (Master's thesis, University of Waterloo). Retrieved from `http://hdl.handle.net/10012/4426`

Grothe, M. (2014). *Never let a fool kiss you or a kiss fool you*. New York: Penguin Books. Retrieved from `https://www.drmardy.com/chiasmus/book` (OCLC: 883328252)

Harris, R., Di Marco, C., Ruan, S., & O'Reilly, C. (2018, 06). An annotation scheme for rhetorical figures. *Argument & Computation*, *9*, 1-21. doi: 10.3233/AAC-180037

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735

Hromada, D. (2011). Initial experiments with multilingual extraction of rhetoric figures by means of perl-compatible regular expressions. In *Proceedings of the second student research workshop associated with ranlp* (pp. 85–90).

Hutchins, W. J. (2004). The georgetown-ibm experiment demonstrated in january 1954. In R. E. Frederking & K. B. Taylor (Eds.), *Machine translation: From real users to research* (pp. 102–114). Springer Berlin Heidelberg. (From the Conference of the Association for Machine Translation in the Americas) doi: 10.1007/978-3-540-30194-3_12

Hutchins, W. J. (2007). Machine translation: A concise history. *Computer aided translation: Theory and practice*.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 655–665).

Kamath, U., Liu, J., & Whitaker, J. (2019). *Deep learning for nlp and speech recognition*. Springer.

Kenton, J. D. M.-W. C., & Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacl-hlt* (pp. 4171–4186).

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2022). Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, 1–32. doi: 10.1007/s11042-022-13428-4

Kumar, E. (2013). *Natural language processing*. IK International Pvt Ltd.

Kučera, H., & Francis, W. N. (1963-64). *Brown university standard corpus of present-day american english.*

Lapa, V. G., & Ivakhnenko, A. G. (1967). *Cybernetics and forecasting techniques* (Vol. 8). American Elsevier Publishing Company.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Malina, J. (2017). *Un general speeches.* data.world. Retrieved 2023-01, from `https://`

data.world/jmalina/un-general-speeches/

Manning, C. (2000). *Probabilistic models in computational linguistics.* Stanford Natural Language Processing Group.

Meyer, Y. (2023). *Application of comprehensive and responsible machine learning methods to the detection of chiasmi's rhetorical salience* (Unpublished master's thesis). University of Passau & INSA Lyon.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 26). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Mitrović, J., O'Reilly, C., Mladenović, M., & Handschuh, S. (2017, 10). Ontological representations of rhetorical figures for argument mining. *Argument & Computation*, *8*(3), 267–287. doi: 10.3233/AAC-170027

Nakayama, H., Kubo, T., Kamura, J., Taniguchi, Y., & Liang, X. (2018). *doccano: Text annotation tool for human.* Retrieved from https://github.com/doccano/doccano

Nallapati, R., Zhou, B., dos Santos, C., Gulcere, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th signll conference on computational natural language learning* (pp. 280–290).

Nayak, P. (2019, oct). *Understanding searches better than ever before.* Retrieved from https://blog.google/products/search/search-language-understanding-bert/

OpenAI. (2022, nov). *Chatgpt: Optimizing language models for dialogue.* Retrieved from https://openai.com/blog/chatgpt/

Pungas, T. (2017). *A dataset of english plaintext jokes.* GitHub. Retrieved 2023-01, from https://github.com/taivop/joke-dataset

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020, July). Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 101–108). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.acl-demos.14 doi: 10.18653/v1/2020.acl-demos.14

Ramesh, D., & Sanampudi, S. K. (2022, March). An automated essay scoring systems: a systematic literature review. *Artificial Intelligence Review*, *55*(3), 2495–2527. Retrieved 2023-01-19, from https://link.springer.com/10.1007/s10462

-021-10068-2 doi: 10.1007/s10462-021-10068-2

Reynolds, C. (1952). *The conference on mechanical translation.* Massachusetts Institute of Technology.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Ruizendaal, R. (2017, 07). *Deep learning #4: Why you need to start using embedding layers.* Retrieved 01-2023, from https://towardsdatascience.com/deep-learning-4-embedding-layers-f9a02d55ac12

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323*. doi: 10.1038/323533a0

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., ... Wolf, T. (2022). *Bloom: A 176b-parameter open-access multilingual language model.* arXiv. Retrieved from https://arxiv.org/abs/2211.05100 doi: 10.48550/ARXIV.2211.05100

Schneider, F., Barz, B., Brandes, P., Marshall, S., & Denzler, J. (2021, November). Data-driven detection of general chiasmi using lexical and semantic features. In *Proceedings of the 5th joint SIGHUM workshop on computational linguistics for cultural heritage, social sciences, humanities and literature* (pp. 96–100). Punta Cana, Dominican Republic (online): Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.latechclfl-1.11 doi: 10.18653/v1/2021.latechclfl-1.11

Seymour, T., Frantsvog, D., & Kumar, S. (2011). History of search engines. *International Journal of Management & Information Systems (IJMIS)*, *15*(4), 47–58.

Sondheimer, N. K. (1987). *The rate of progress in natural language processing* (Tech. Rep.). Marina del Rey, CA 90292: University of Southern California, Information Sciences Institute.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 27). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

Turing, A. M. (1950, 10). Computing Machinery and Intelligence. *Mind*, *LIX*(236), 433-460. doi: 10.1093/mind/LIX.236.433

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran As-

sociates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wachsmuth, H., Al-Khatib, K., & Stein, B. (2016, December). Using argument mining to assess the argumentation quality of essays. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 1680–1691). Osaka, Japan: The COLING 2016 Organizing Committee. Retrieved from https://aclanthology.org/C16-1158

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Rush, A. (2020a). Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38–45). Online: Association for Computational Linguistics. Retrieved 2023-01-19, from https://www.aclweb.org/anthology/2020.emnlp-demos.6 doi: 10.18653/v1/2020.emnlp-demos.6

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Rush, A. (2020b, October). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.emnlp-demos.6 doi: 10.18653/v1/2020.emnlp-demos.6

Yadav, A., & Vishwakarma, D. K. (2020, August). Sentiment analysis using deep learning architectures: a review. *Artificial Intelligence Review*, *53*(6), 4335–4385. Retrieved 2023-01-19, from http://link.springer.com/10.1007/s10462-019-09794-5 doi: 10.1007/s10462-019-09794-5

Appendix

A List of Acronyms

Acronym	Meaning
NLP	Natural Language Processing
PoS	Part-Of-Speech
ML	Machine Learning
SVM	Support Vector Machine
DL	Deep Learning
TL	Transfer Learning
ANN	Artificial Neural Network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory

B List of Definitions

Linguistic definitions:

- **Chiasmus**: The inverse repetition of any two pairs of linguistic elements in a larger coherent body of text.
- **Antimetabole:** A "lexical chiasmus", that is an inverse repetition of lexical elements named "lemmata".
- **Lemma (plural lemmata or lemmas):** In linguistics, the canonical form of a given set of related words.

Machine Learning definitions:

- **Machine Learning (ML)**: "The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data."
- **Classical Machine Learning:** The set of statistical tools aiming to learn conclusions about a given dataset by using specific mathematical features, like the distance between data-points or the direction of vectors.
- **Artificial Neural Networks ("ANN"):** A network of interconnected nodes which aims to emulate a simplified version of animal brains to process inputs: the nodes are therefore called neurons, and the whole network a neural network.

- **Deep Neural Network ("DNN")**: An ANN with a high number of neuron layers between the input and the output, usually more than ten.
- **Deep Learning ("DL")**: The training and use of Deep Neural Networks to execute tasks and solve problems.
- **Transformer**: A Deep Neural Network which uses mechanisms of self-learned attention to process entire input sequences in a parallelised, permutation invariant manner.

C List of Figures

Eidesstattliche Erklärung

Ich versichere hiermit wahrheitsgemäß, die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung der Universität Passau zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet zu haben.

Passau, den 19.01.2023 _____

Guillaume Berthomet