Code: 825 Année universitaire: 2019-2020

Université de la Manouba

Rapport de mémoire

Présenté en vue de l'obtention du diplôme de
Mastère de recherche en Web Intelligence

Sujet

Learning better from low-resource languages: A new approach

Élaboré par:
Manai Elyes

Encadré par

INSAT **Mme.Lilia SFAXI**
InstaDeep **Mme.Nourchene FERCHICHI**

Abstract

The amount of online text content is exponentially growing by the minute. This has led to the evolution of a field in computer science called Natural Language Processing (NLP) which aims to make computer understand human speech text and extract useful information from it. The most recent advances in NLP has led to the use of Artificial Intelligence (AI), more specifically Deep Learning (DL) to make models that can learn and understands languages from huge amounts of text that also perform really well on text tasks such as classification, summarization and translation. The most recent models have even beaten the human scores.

In this thesis, we study the how well these state-of-the-art (SOTA) AI models, that usually need huge amounts of text to perform well, fare on small datasets. we also explore ways to augment the small dataset and see how well models that have been trained on these augmented datasets perform versus the base models and versus models trained on big datasets. This is because aside from the popular languages like English, French, Arabic, German etc. low resource languages, like dialects, endangered languages or even some national languages, don't have the needed amount of online text content available. This greatly limits the number of people that can benefit from these SOTA models, which goes against our belief that the key to more progress is the democratization of technology.

In this context, we propose a new methodology called **Pre-training Data Augmentation (PDA)** that uses data augmentation to help train small dataset using sophisticated models. We show that, overall, our approach improves the learning results of NLP language models on small datasets (100 Megabytes or less) as well as their fine-tuning results on downstream tasks.

Acknowledgements

I would like to start by thanking InstaDeep for accepting to let me work on my own research ideas in their environment and for being understanding and supportive. I particularly want to express my gratitude to Miss Lilia Sfaxi and Miss Nourchène Ferchichi for their dedication and efforts in making this project successful. This work wouldn't have seen the light of day without their mentoring.
Last but not least, I would like to thank the members of the jury for putting in the effort to read and evaluate my work. Your feedback is very important to me.

Dedications

I wish to offer my deepest thanks to all those who helped me throughout this Journey: My academic supervisor Miss Lilia Sfaxi that had to put up with all my nonsense and “bright ideas” and allowed me to focus on the work at hand. My professional supervisors Miss Nourchène Ferchichi and Ahmed Cheikhrouhou that had to keep up with my incessant rants and help me each time I break a company computer, the Coronavirus for letting me stay at home and focus only on this thesis, the authors of the papers and libraries my work is based on that I kept spamming with emails of me finding bugs and finally my university that was very tolerant of my circumstances and made exceptions for me.

Table of Content

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
ANN	Artificial Neural Networks
RNN	Recurrent Neural Networks
CAWE	Contextual Augmentation with Word Embeddings
BERT	Bidirectional Encoder Representations from Transformers
biLM	bidirectional Language Model
TF-IDF	Term Frequency divided by Inverse Document Frequency
LM	Language Model
LSTM	Long Short-term Memory
GRU	Gated Recurrent Unit
GloVe	Global Vectors for Word Representation
ELMo	Embeddings from Language Models
LSA	Linguistic Society of America
QoD	Quality of Data
HBM	Heuristics Based Methods
MBM	Models Based Methods
MM	Mixed Methods
SMA	Single Model Architecture
MMA	Multiple Model Architecture

General Introduction

Technology has now become mankind's biggest revolution. It allowed humans to enhance work productivity, relax more, dream more and give a chance to those who had a hard time integrating society. The Artificial Intelligence field, in particular, experienced an exponentially high spike in interest in the past decade. This is due to both the increasing availability of online Data (also known as Big Data) and the increase in computational power. This progress in Artificial Intelligence (AI), and especially Machine Learning (ML) and Deep Learning (DL), has been so powerful that it is now in almost all fields around us, from our smartphones to our cars. Natural Language processing, a particular field in Artificial Intelligence, has seen an enormous boost in capabilities.

1 Motivation

Humans have already been using NLP-based apps like smartphones, GPS, voice assistants and chatbots for decades. As we humans communicate through natural language, having computers understand our intents can result in very practical use cases like Text Summarization and Question Answering. Up until some years ago, most NLP based software used rule-based logic because Machine Learning wasn't efficient enough yet, since they lack the data. With the exponential increase of online data, that is not a problem anymore, and the NLP community has since become more and more interested in the field.

As we can see in Figure 1, NLP has just broken past its peak inflated interest and moved on to the next phase, meaning that we now know how to get general results and are now going towards the path of having specialized NLP applications depending on the very specific needs of each use case.

It is through this progress that we can now talk about having chatbots in any dialect we want, programs that can summarize books, and voice assistants so real that one sometimes can't distinguish if they're AI or human. NLP is a very interesting field to us because it allows us to study how the human brain understands language, find patterns or models, then try to simulate those patterns in code. It is as if we are discovering the underlying behavior of our minds and it is for that reason that we are interested in it. A second point of interest is that NLP can solve a lot of real life problems, voice assistants for example are lifesavers for blind people. Automatic

[1] Hyper Gartner Cycle for Artificial Intelligence, 2019, https://www.forbes.com/sites/louiscolumbus/2019/09/25/whats-new-in-gartners-hype-cycle-for-ai-2019/#352e10a8547b

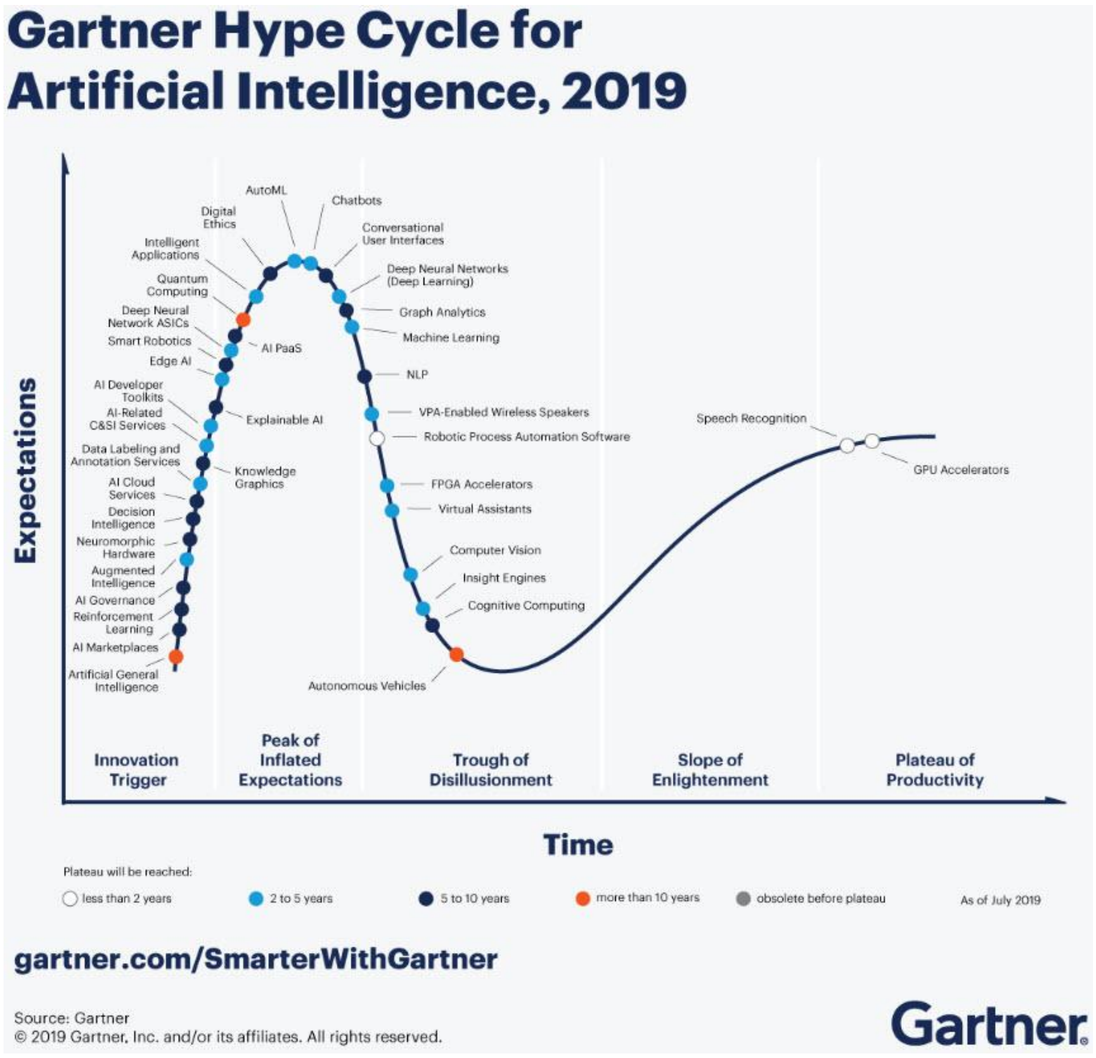


Figure 1 – Hyper Gartner Cycle for Artificial Intelligence 2019 [1]

translators are lifesavers for people stranded in foreign countries. Chatbots ease a lot the pain of community managers and small business owners. Sentiment analysis helps individuals and brands better understand their fans and react to them. In the middle of the Coronavirus, a surge of chatbots appeared that would answer people's questions. This eased a lot the stress of tele-operators, lowered the stress of the telecommunications infrastructure by needing less calls, and helped people stay informed and safe. However, we observed that while English and French speaking chatbots were rapidly deployed, there was a lack of chatbots in other languages. We saw several groups share their in-construction chatbots asking their networks to talk to it and feed it information in their local dialects. We never saw these chatbots again for the rest of the pandemic. We can assume that their chatbots didn't get good enough to be deployed. This poses a serious question: Can only popular languages exploit the current power of NLP?

Just how much data does one need to make a reasonably good NLP model in a “low resource” language? There are thousands of dialects and languages in the world for which we can only find small portions of publicly available content. The best places to find content right now are social media, but then we’ll touch the sensitive domain of data privacy.

2 Research questions & objectives

Building on the previous section, this thesis will be examining the most recent and successful NLP Machine Learning models and the data it’s trained on, to study how successful they are or would be on low resource languages. We will also be looking at how to make the training of NLP ML models better on these small datasets. More specifically, we will be looking at how to make the most out of the available data we have, on the basis that in the low resource languages regime, we don’t really have a say in what data we’ll manage to get. We will therefore have the added constraint of coming up with a method that can work on any target language out of the box. Since the general rule of thumb is that the more data we have, the better our models will be, we already have an assumption that we’ll test. This assumption is based on the concept of Data Augmentation, a successful method that’s been used for a long time in Computer Vision [1–6] and started being implemented in NLP. We will see if there is a way to use or alter these Data Augmentation principles to fit our needs. To conclude, we have two main objectives that we will investigate in the course of this master thesis:

1. How do state-of-the-art NLP models work and why do they get such good results?
2. Can Data Augmentation help NLP models learn better from low-resource languages?

3 Thesis structure

This thesis report is divided in five chapters. Chapter one provides a theoretical explanation of the most important concepts in Artificial Intelligence and Natural Language Processing and ends with our problematic. Chapter two reviews the state-of-the-art works relevant to our problematic and a thorough analysis of them. Chapter three provides a detailed explanation of our proposed methodologies to augment the initial dataset. Chapter four discusses the results of our methods and a comparison to other models trained on more data. Chapter five will present some interesting findings we stumbled upon and the different improvement ideas that came to our minds while experimenting. Lastly, we will give the conclusion of the project and discuss some final recommendations for further improvements.

Chapter I

Theoretical Background

This chapter will focus on giving the reader the right background knowledge to easily follow through the rest of the chapters. Due to the enormous amount of details one can get in, we will only focus on the concepts that are most important to us in the context of this thesis.

1 Fundamentals of Artificial Intelligence

This section will briefly take us through the difference between Artificial intelligence, Machine Learning and Deep Learning. While these terms are usually used interchangeably, they do hold differences and that's what we'll be explaining.

1.1 Artificial Intelligence (AI)

Artificial Intelligence is a subset of the field of Computer Science that focuses on making computers intelligent. It started from the 60s with machines that try to converse like humans and do tasks that were meant for humans. The early stages of Artificial Intelligence were lists of rules and logic implemented by developers in big systems that would later be called expert systems.

1.2 Machine Learning (ML)

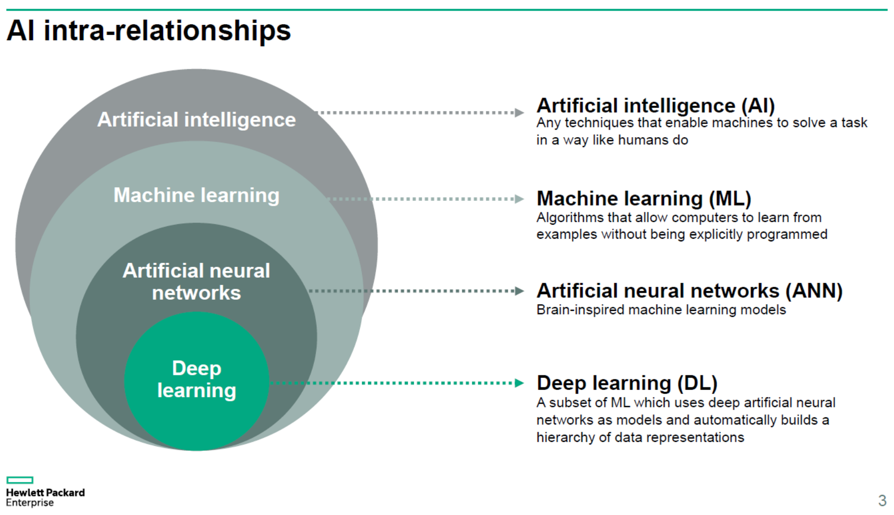
Machine Learning is a subset of Artificial Intelligence where the machine has to learn rules from the data itself without being explicitly told and programmed to do so by the developers. Machine Learning programs are based on mathematical equations that can learn mappings from given inputs to given outputs. This allows them to find clues that humans may miss and thus get better results at whatever tasks they are trained on. Examples of ML methods are K-Nearest Neighbors [7] and Support Vector Machines [8].

1.3 Deep Learning (DL)

Deep Learning is a subset of Machine Learning that relies only on Deep Neural Networks [9], a revolutionary architecture that can be trained to deal with complex equations and problems.

Deep Learning was especially proven to be efficient when dealing with Images, Videos and text which were very hard to deal with using Machine Learning algorithms. However, by using Neural networks, we are sacrificing explainability for results, and it is therefore very hard to understand why our models learned something, which makes it very hard to improve one's work. It is however on this principle that the currently best NLP are based on. We will therefore be referring to Deep Learning based models for the rest of the thesis.

Figure I.1 summarizes the relationship between AI, ML and DL.



Artificial intelligence (AI)
Any techniques that enable machines to solve a task in a way like humans do

Machine learning (ML)
Algorithms that allow computers to learn from examples without being explicitly programmed

Artificial neural networks (ANN)
Brain-inspired machine learning models

Deep learning (DL)
A subset of ML which uses deep artificial neural networks as models and automatically builds a hierarchy of data representations

Figure I.1 – AI intra-relationships[1]

2 Fundamentals of Natural Language Processing (NLP)

Since Natural Language Processing is a big and diverse field, we will briefly introduce it again, give a short summary of how it progressed through the years and present the current state-of-the-art solutions.

2.1 NLP definition

Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to make computers as human as possible. NLP thrives to do that through understanding text data.

[1]The yin and the yang of AI and Blockchain, 2018, https://blog.3g4g.co.uk/2018/10/the-yin-and-yang-of-ai-blockchain.html

Text is an unstructured data type that varies a lot from source to source. Examples are books, papers, articles, social media messages and comments, which can vary in manner of speech, language, length and contextual meaning.

We can exploit this data to make computers perform well on tasks like Text Classification, Question Answering, Language Modeling, Sentiment Analysis, Topic Modeling etc. Yet, each task has a different and specific preprocessing and data selection process to optimize the results.

It is thus the combination of the randomness of the data with the specific needs of the NLP tasks that makes NLP not easy and not straightforward. It also is one of the fields that didn't have any standard go-to methods until very recently (we'll explore this point in the next sections). In the next sections, we will explore the different ways NLP was dealt with. We will group these methods in 3 categories: Classical NLP, Machine Learning NLP and Deep Learning NLP.

2.2 NLP with classical methods

Before Machine Learning became popular and as efficient as it currently is, the NLP community used heuristics and logic-based methods to deal with NLP tasks, using (but not limited to) Label Encoding, TF-IDF [10], and WordNet [11]. We focus in the following part on these three famous techniques.

2.2.1 Label Encoding

Label encoding is the act of assigning unique numbers to words (see Table I.1). The assigning in itself can be tweaked or programmed in a way to group similar words together based on characteristics like length, meaning, context, rarity etc. This makes programming software that deals with text faster and easier as integers not only take less space in RAM but also needs less typing and cluster in the code as developers don't need to use words in their code anymore.

Tableau I.1 – Basic Example of Label Encoding of a 9847-words text

Original word	**you**	we	they	sleep	...	cat	dog	bird
Encoded	**0**	1	2	3	...	9845	9846	9847

This method however requires a lot of human interaction, verification and correction, all of which are time consuming, tedious and prone to errors.

2.2.2 TF-IDF

Used by search engines for many years, the TF-IDF [10] method is based on an equation (see figure I.2, left side) that calculates the rarity of words across a number of documents. It then assigns a value for each word in the documents that represents its rarity across them. As seen in Figure I.2 (right side), a very low value means that it's a stop word and can be ignored, a very high value means it is a rare word, and in between will be normal but useful words. We can then use these values for grouping the words by the usefulness of the information they convey: stopwords would, for example, be removed while the rest could be processed in a way that allows us to, for example, deduce the topics of the source documents. When used across many documents, we can use TF-IDF to measure similarity between them, which is why it is used by search engines.

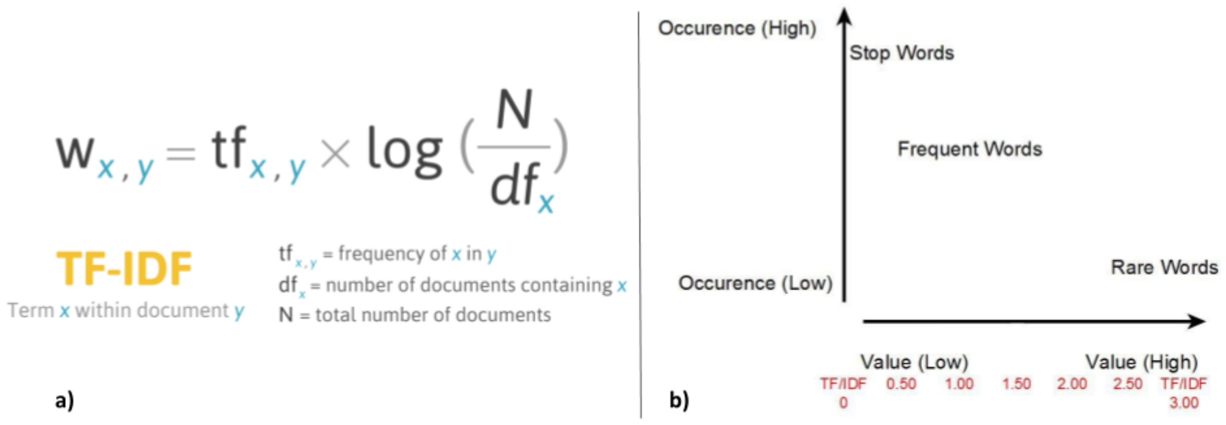

$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$
TF-IDF
Term x within document y
$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents
a)
Occurence (High)
Stop Words
Frequent Words
Occurence (Low)
Rare Words
Value (Low)
Value (High)
TF/IDF 0
0.50
1.00
1.50
2.00
2.50
TF/IDF 3.00
b)

Figure I.2 – The TF-IDF equation[2](left) and term weight representation[3](right)

This method, as easy and fast as it is, doesn't however give us information about the relationships between words, like synonyms, and is thus limited in where it can be applied.

2.2.3 WordNet

A team in the Princeton University's Psychology Department came up with the idea of porting the power of graphs to NLP, and made WordNet[11]: A graph of the English language containing synonyms, antonyms, homonyms etc. This graph helped NLP practitioners make better and more complex programs like writing assistants, proofreaders and auto-correct with just a couple of lines of code.

[3]The TF-IDF equation, https://www.quentinfily.fr/tf-idf-pertinence-lexicale/
[3]term weight representation, https://towardsdatascience.com/tfidf-for-piece-of-text-in-python
[4]WordNet, https://openscience.com/wordnet-open-access-data-in-linguistics/

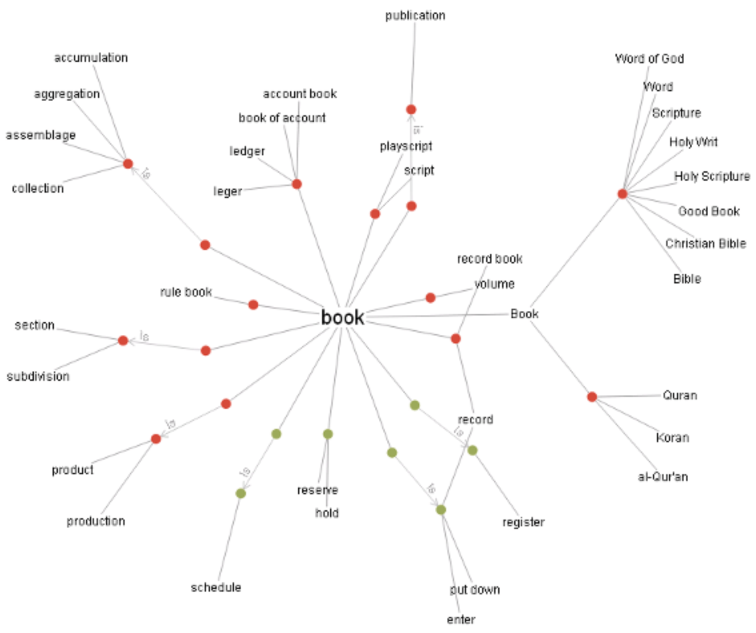


Figure I.3 – Glimpse of what WordNet returns when given the word "Book"[4]

The idea of Making a graph was very ingenious but making one and updating it both take a lot of time and effort, especially with the rapid expansion of the human vocabulary.

Conclusion

Even with just classical methods, NLP researchers managed to make successful representations of text (labels, weights, graphs), extract Lexical Syntactic Information (weights, counts. . .) and model Lexical Relationship (family, hypernyms, entities etc. through WordNet). However, these methods can't capture semantic relations between words which greatly limits its application.

2.3 NLP with Machine Learning

Machine Learning's self-learning capabilities caught the eye of pioneers in the NLP community that identified, very early, its potential in natural language. Several solutions exist, we present in this part the most famous one, Word2Vec [12], that managed to make a breakthrough.

2.3.1 Word2Vec

In 2018, Mikolovs et al's[12] paper introduced Word2Vec which was the breakthrough of ML in NLP. Word2Vec used a combination of one hot encoding (transforming the input text into a binary list) and shallow neural networks (see figure I.4) to achieve astonishing results on a task called Language modeling, more known as auto-complete, that suggests the next word given your current sentence. Trying to do this without ML would either require a huge amount of effort for a small use case, or plain bad results.

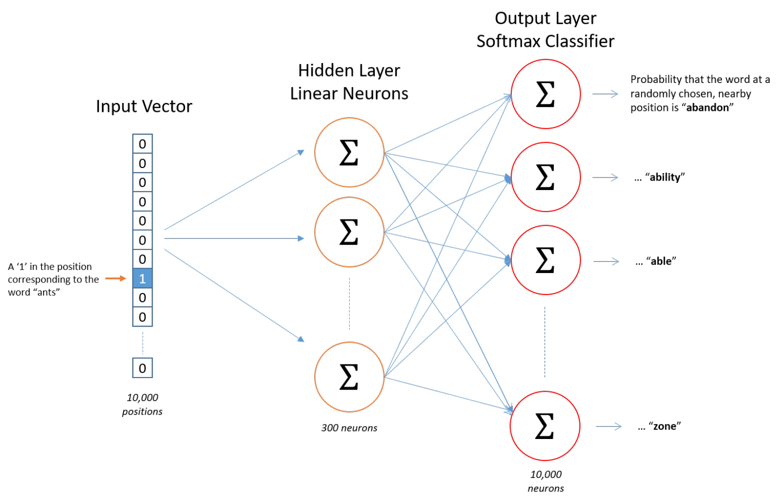


Figure I.4 – The Word2Vec Architecture[5]

This allowed Word2Vec to learn a very good representation of words called word embeddings in the form of dense vectors. These word embeddings are the reason why Word2Vec managed to become a reference in the NLP world. In short, they gave us a way to use math on text and we can therefore perform tasks like similarity detection between words and paragraphs without prior help from humans (like WordNet). Figure I.5 illustrates what word embeddings look like and how we can, for example, use them to calculate word distances.

The amazing results Word2Vec gave sparkled worldwide interest, and from then on, came more and more ML based NLP models like Glove[13] and FastText[14], each covering the previous one's shortcomings.

Conclusion

By applying Machine Learning to NLP, researchers managed to manipulate text way more efficiently by capturing semantic relationships between words. The most interesting part of this being that it is achieved in an unsupervised manner that allows the same technique to be used out of the box on any dataset. The main issue left to deal with is the lack of context awareness as embeddings only take into consideration a small window of context.

[5]Word2Vec Architecture, http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-m
[6]Word embeddings, https://medium.com/@hari4om/word-embedding-d816f643140

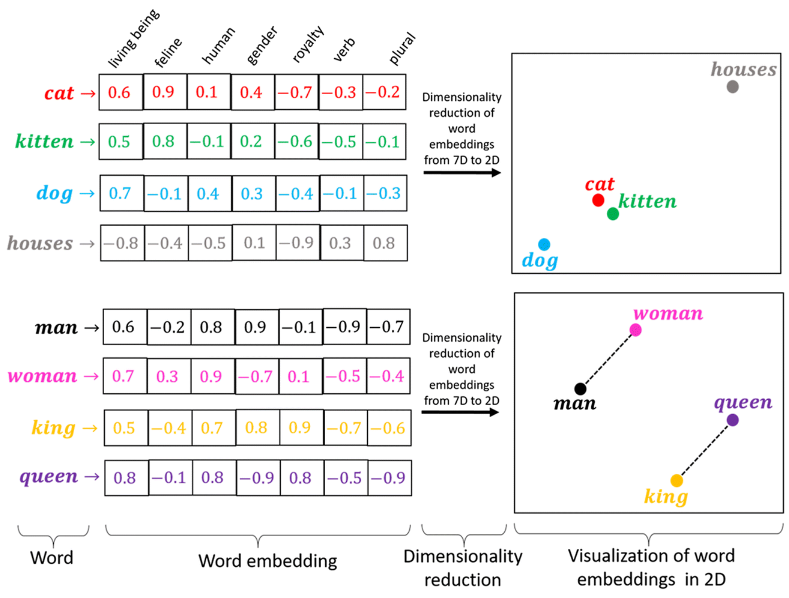

living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Figure I.5 – Word Embeddings (left) and their projections on a 2D space (right), creating a notion of word distances[6]

2.4 NLP with Deep Learning

With the continuous success of Deep Neural Networks over classical methods and plain Machine Learning algorithms, it should come to no surprise that the NLP community decided to try it out. In the next sections we will explore how NLP transitioned to Deep Learning.

2.4.1 Single Model Architectures (SMA)

Since ML NLP is based on shallow neural networks, it can only process 2 words at a time. Yet text is comprised of long sentences and paragraphs, so ML based solutions started breaking down and giving bad results when given long text sequences. To counter this, the use of multi layered neural networks started being tested, resulting in Recurrent Neural Networks [15] (RNN) (see figure I.6) that could indeed process long sequences of text at a time without losing information.

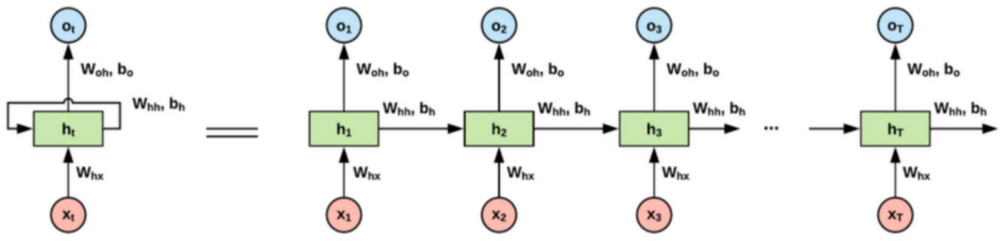


Figure I.6 – The RNN architecture[7]

In short, RNNs work like Word2Vec, but this time, it suggests the next word given more than 1 or 2 previous words. Figure 7 gives us a glimpse of how it works; to suggest the n-th word of a sequence, the n-1 hidden states (weight vectors representing the words given the context) are averaged together in a sequential manner to keep information of the whole sequence. The final hidden state will have the combined information of the sentence and will be used as final input to predict the next word. The RNN-based Neural Networks have been the winners of NLP race for many years. Of the many variations in RNN-based Neural Networks, the two most successful architectures are LSTMS[16] and GRUs[17]. As the figure I.7 tells, the variations happen at the architecture level of the models.

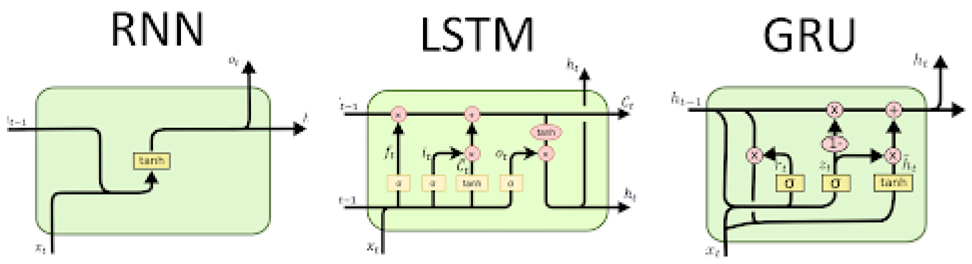


Figure I.7 – RNN variations: Plain RNN, LSTM and GRU (left to right)

2.4.2 Multiple Model architectures (MMA)

While SMAs have a great learning potential, they only learn in one direction: either left to right or right to left. To illustrate this, in figure I.8, we can see that the word "bank" has multiple meanings depending on the context it is in, more specifically, depending on the direction of the context, its meaning can differ a lot.

Left to Right: Bank ___ → deposit, transfer, heist

Right to Left: ___ bank → river, big, World

Figure I.8 – Example of context changing the meaning of the word "bank"

To counter this, researchers came up with the idea of combining two models, each learning in a direction, and averaging the weights of a word in each model to get the final, general word embedding. In this category, we can mention Bi-LSTMs [18] and ELMO [19] as very efficient examples of bi-directional RNN Models.

[7]RNN architecture, https://towardsdatascience.com/introduction-to-recurrent-neural-network
[7]RNN variations, http://dprogrammer.org/rnn-lstm-gru

2.4.3 Pre-training

NLP training before 2018 was mostly supervised, requiring labeled datasets. Yet, getting labeled data is expensive, tedious and time-consuming, basically forming an obstacle. In, 2018, a paper called ULMFIT [20] has introduced the concept of transfer learning [21] - which was common in Computer Vision [22] - to NLP. Transfer learning is the act of using an already trained model as base model instead of starting from scratch, thus needing less data and time to get good results on specific use cases.

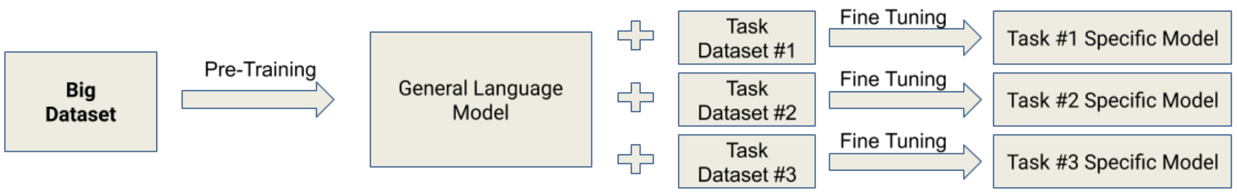


Figure I.9 – Full Transfer Learning Pipeline

As seen in Figure I.9, Transfer learning therefore constitutes of two main steps: pre-training and fine-tuning, with pre-training being the act of training a model from scratch on a general task, and fine-tuning being the act of taking a trained model as a base and further train it on our own data and tasks. This generally gives very good results for a fraction of the usually needed data. The reason why it works so well is because there are shared representations even between different data.

Conclusion

Deep Neural Networks managed to get NLP to a whole new level as Deep Learning based NLP models have been consistently beating plain Machine Learning based NLP models. Since 2018, Transfer learning with MMAs has become the main method of dealing with NLP tasks. This however did not last long, as we're going to see in the next section.

2.5 The Transformers invasion

While the NLP world has started to become quiet as progress has become slow, suddenly, a paper was published that would completely change the NLP landscape. This paper introduced the concept of Transformers [23], and from then on the NLP word went back into full speed. We will explore exactly what Transformers and why they disrupted NLP research.

2.5.1 What are Transformers?

Transformers [23] are a recently published Neural Network architecture that greatly outperformed all previous work and became the go-to models for NLP. Published in 2018, the "Attention is all you need" paper [23] introduced a new concept called attention accompanied with a new architecture called Transformer (see figure I.10). Not only does this architecture not need multiple layers like RNNs, but it could also process sequences in parallel, thus making it finally possible to exploit the power of parallelization and speed up training time.

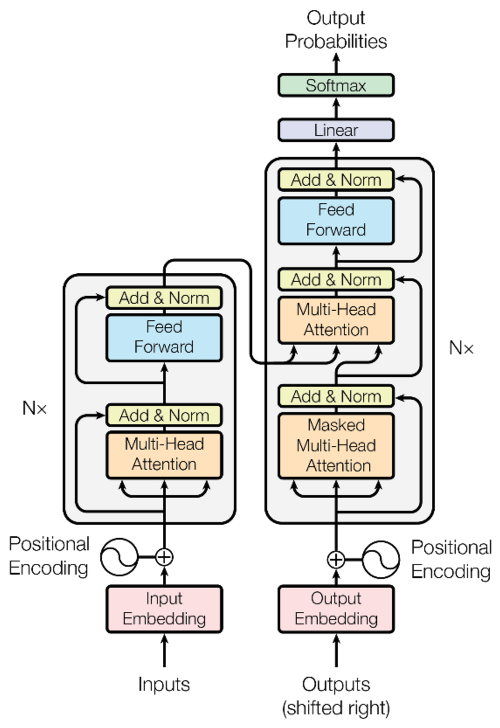


Figure I.10 – The Transformer architecture[8]

It didn't take long for the NLP community to start testing this model out. In 2018, OpenAI introduced GPT [24], the first publicly available large scale Transformer that also greatly outperformed state-of-the-art models at the time on a multitude of tasks like summarization, language generation and question Answering, thus displaying the power of this architecture. Not even a year later, they published a better version, GPT-2 [25]. Google then published their own variation, called BERT [26], that outperformed even GPT-2,only to then be dethroned by XLNET [27] a few months later, only for this last one to also be dethroned by RoBERTA [28]. This kept going since 2018, creating general curiosity and interest, and serving as the waking

[8]Transformers Architecture, https://www.researchgate.net/figure/The-Transformer-model-architecture_fig1_323904682

call that would shift the attention of the whole NLP community and start the Transformers Race.

2.5.2 The Transformers race

In the span of only 2 years, Transformers went from being nonexistent to monopolizing the NLP world, with no less than 17 variations (as of the day of writing this report), all outperforming each other. We can especially see a trend of tech giants (Google, Facebook, Microsoft) investing a lot in Transformers and going toe to toe against each other.

In figure I.11, we can see how fast progress is going into publishing new models. The x-axis denotes the year of publication and the y-axis denotes the number of parameters in Millions.

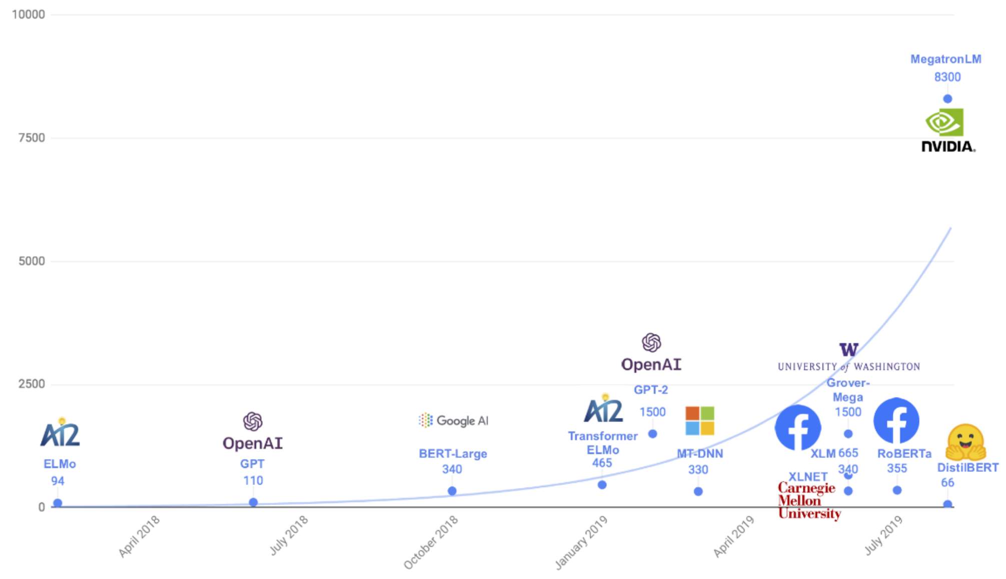


Figure I.11 – Transformers models publishing timeline[9]

This growth is to be expected, since they kept crushing previously unbeatable NLP benchmarks such as GLUE[29]. As we can see in Table I.2, we can see how far the Transformers based models (ranks 1 to 32) are in score to RNN-based models (ranks 33 and below). Not only that, we can also see that the least efficient Transformer model still performs 5 points better than the best RNN model. This just shows how powerful Transformers are.

[9]The Transformers race, https://towardsdatascience.com/2019-the-year-of-bert-354e8106f7ba

Tableau I.2 – Most recent rankings on the GLUE benchmark

Rank	Model	Score
1	ALBERT [30] + DAAF + NAS	90.5
2	ERNIE [31]	90.4
3	StructBERT [32]	90.3
4	T5 [33]	90.3
...	...	...
32	EL-BERT	75.6
33	BiLSTM + ELMo + Attn	70.0
...	...	...

2.6 The problem with Transformers

Just like everything in this world, Transformers have pros and cons. Two of these cons stick out by a big margin: the size of the models and the need for data. As seen in figure I.11, the size of the models (y-axis) is getting bigger and bigger, and training these behemoths has become nearly impossible for independent researchers and small labs. This is also why in the previous section we pointed out that it's only the tech giants that are competing in this space; they're the only ones with access to enough resources to do so. Even fine-tuning takes a lot of time on standard hardware, which is why there have been a lot of efforts in making pre-trained models smaller so that fine-tuning them would take a lot less time. Size, however, is not our concern. In the context of this thesis, we're more interested in the second con: the need for data.

Tableau I.3 – Characteristics of popular Transformers models

Model	Language	Number of Parameters	Pre-training dataset size
BERT Small [26]	English	110M	13G
BERT Large [26]	English	340M	13G
AraBERT [32]	Arabic	340M	24G
FlauBERT [33]	French	340Ms	71G
XLNET [27]	English	340M	170G
T5 [33]	English	220M	750G
GPT-2 [25]	English	1500M	40G

In table I.3, we can see the pre-training dataset sizes for different models in different languages. As we can see, the smallest dataset size is 13G. Transformers are hungry for data, and it has now been proven[34, 35] that the more data you give them the better they become, and this is exactly our problematic. The size of the data is maybe the biggest obstacle in getting good NLP Models, and dealing with this is exactly our main research objective.

3 Problematic & Research objectives

Now that we are more familiar with what's happening in the NLP field, we can see that we shifted from "what can we do?" to "what can we not do?". As optimistic as it sounds, it's not for everyone yet as you can only say that when you have a lot of pre-training data. We'll discuss in the next sections where the problem lies and why it is a problem.

3.1 Problematic: What about low resource languages?

A gigabyte of text is a lot. While the newest models are getting us good results, they're only being trained on popular languages with easily available data online. Yet there are thousands of local languages, dialects and low resource languages spoken. According to the Linguistic Society of America (LSA)[36], there are 6909 distinct Languages in the world as of 2010. The Endangered Languages Project (ELP)[10] counts 3429 endangered languages (less than 1000 speakers). These languages and dialects do not have gigabytes of online data that we can scrap and use, and thus, exploitation of the power of transformers is greatly limited. During COVID-19 for example, while English, French and Germans chatbots were deployed to help tele-operators and keep the public up to date, Tunisians, for a example, were struggling to get volunteers to chat with prototype chatbots to fill them in with data. This behavior isn't limited to Tunisians, but locals all over the world couldn't exploit the technology. We believe in the democratization of technology, and that in the future, everyone should be able to exploit the power of AI. The work presented in this thesis is therefore our contribution to bridge that gap.

3.2 Research Objective: Towards universal access to Transformers

The objective of this work is to see how well Transformers models fare on small datasets, and how we can make them learn better and get acceptable results on downstream tasks. We will explore if there are ways to make our small starting datasets bigger and/or better and will try to discover more about how we can make the most out of small starting data to get the best results.

Conclusion

We presented in this chapter the most important concepts of NLP and how it evolved through the years. We also introduced the state-of-the-art NLP models and highlighted the biggest

[10]Endangered Languages Project (ELP), last consulted in June 2020, http://www.endangeredlanguages.com/

challenges in exploiting them which is the need for huge amounts of data. In the next section, we will explore what other researchers did to overcome this limitation.

Chapter II

State of the art

In this chapter, we will explore the literature and discover what researchers worldwide have managed to uncover that could help us spot an opportunity to focus on.

1 The research direction of Transformers

Until now, NLP research using Transformers has been, for the most part, about getting better results in common benchmarks like GLUE by making bigger models and using more data instead of going for less data[25–28, 33–35].
In accordance to that, and to the best of our findings, we did not find papers directly related to our use case. There was only one work that is vaguely similar that we will discuss at the end of this chapter. We did however find a lot of papers that are indirectly helpful, even though their objectives aren't the same as ours.

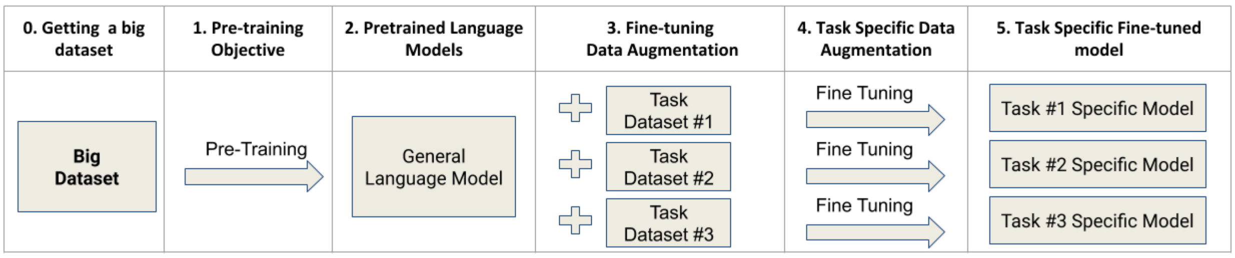


Figure II.1 – The full pipeline of training Language Models

Figure II.1 shows us again the full pipeline of training Language Models, this time with the steps numbered and separated. The related works in this section are divided following this separation.

2 Review of related works

There's been quite a number of research works that target Language learning. We'll be exploring those that mattter the most to us.

2.1 Pre-training objective

The research work in this category focused on the way the Language Models learn in the pre-training phase. More specifically, they focus on how to model the learning problem and how to present the learning task. For example, while the most popular language model BERT [26] is based on "next word prediction" and "Next sentence classification", RoBERTa [28], which gave better results than BERT, decided to drop the latter and focus on the prior while providing a lot more training data. XLNET [27] decided to go with randomly swapping the words in a sequence and making the model predict the next word. ELECTRA [37] downright changed the objective of the learning by adding an adversarial aspect to it, and the model, which is based on a BERT model, now calculates the loss of all words given an input sequence, not just the loss of the word to predict.

The work in this category is therefore concerned with the results on benchmarks and training time. For them, the problem of small data isn't a concern since, in cases like RoBERTA, they actually solved the problem by adding even more data.

2.2 Pre-trained language models

The research work in this category focuses on optimizing already trained language models. More specifically, they try to make working with the models easier for the fine-tuning phase. For example, DistillBERT [38] uses the knowledge distillation technique [39] to train a smaller language model using a larger pre-trained model, thus generating a new one that's very small in terms of size yet is as efficient as the original. Poor man's BERT [40] proved that some layers in the trained models can be dropped without relevant loss in performance, and MobileBERT[41] used a mix of Knowledge Distillation, Quantization[42] and adding of linear transformations to make their models as efficient as the original one yet so small they work on mobile devices. The work in this category is therefore concerned with making large pre-trained models smaller in size (less layers, parameters, weights, disk space) and thus faster for both fine-tuning and inference.

2.3 Fine-tuning dataset augmentation

The research work in this category focused on augmenting the fine-tuning datasets. More specifically, they try to make the dataset bigger without adding too much noise so that the model learns well on downstream tasks. For example, Kobayashi[43] introduced the idea of using the word embeddings of language models as a way to replace words with others whose embedding vectors are very similar. Kumar et al.[44] and Wu [45] further improved the idea by using transformers as the generators. Xie et al. [46] augmented their data using back-translation, which is to translate a sentence to other languages and translating them back, using cloud APIs. Dongju [47] added label information as extra details to the trained model so that it augments data while not going too off-track. Yu [48] proposed the use of the Attention mechanism to keep the most important parts of a text and use that information as basis for the augmentations instead of the whole sequence.

An honorable mention to include here is the EDA [49] paper that introduced a set of easy operations like randomly swapping, deleting, replacing and inserting words in a sequence to generate multiple variations of the same input sequence. While the paper wasn't targeting the fine-tuning phase nor working on Transformers, the method can however be used in this context. The work in this category is therefore concerned with making the labeled and task specific datasets bigger and better in order to give the model more valuable information to learn better.

2.4 Task specific data augmentation

The research work in this category also focused on augmenting the fine-tuning datasets, but instead of going for generic methods like the methods in the last section, these found that there are hidden information in the data itself that can be used. For example, Yang [50] observed that in conversations, different answers can be used to answer the same message, and thus used that as a way to find similar sequences and augment the dataset. Gao et al. [51] found that in Neural Machine Translation, replacing a word with a probabilistic distribution, also known as soft word, instead of a discrete word, improves the results in the Machine Translation task.

The work in this category is therefore concerned with making the task specific datasets better, but they also focused on exploiting the intrinsic relationships in the data itself to get even better quality augmentations.

3 Discussion and conclusion

While all the previously mentioned works came up with useful insights and positive results, there is one common flaw: they all assume we have trained or are training on a big pre-training dataset. Since our focus is on languages with little available data, we can't exploit the findings of those papers. We need a way to either make the Transformers learn better on less data or augment the initial dataset. To the best of our knowledge, there has only been one work that tried to make Transformers learn well on little data. In Dai et al. [52], the authors managed to make a Transformer model get better results on the language modeling task, whose scoring is called perplexity (log of the evaluation loss), than RNN based models like LSTMs. The pre-training dataset is the Wikitext-2 dataset [53], a 4 megabytes big text dataset that used to be a popular benchmark for the language learning task before transformers. The results are indeed an interesting feat, but they don't answer our problematic because:

1. The paper only focused on the pre-training phase, while we are interested in the down-stream tasks.

2. The paper only compared against RNN based models, which are outdated at this point, while we want to see how our model can compete with other Transformers .

3. The method they used is a regularization method similar to dropout [54], but that makes the training phase exponentially longer per single experimentation, even on high end GPUs. We don't have that kind of hardware and we prefer to devise a method that can be used by people regardless of their hardware.

As such, we can conclude from the literature review that the current state of the art is lacking feasible and realistic solutions for our problem. This is why we propose our own methodology that works on any initial dataset out of the box, does not require huge amounts of computation or long amounts of time and does well on both the pre-training and fine-tuning tasks. We present this methodology in the next chapter.

Chapter III

Pre-training Data Augmentation (PDA)

This chapter presents our own ideas for the approach we can take to deal with the problem of having too little data. We will first present how we got inspired and will then detail how exactly our plan can be put into place.

1 Background and motivation

Basing ourselves on the “scaling laws of Neural Networks paper” [34] that proved that the more data a model gets trained on the better it performs, and on the previous chapter’s data augmentation work that showed that data augmentation gives better results [43, 44, 46], we came up with the idea of combining both approaches and propose the idea of augmenting the pre-training dataset itself.

Since Language learning is about learning the correct distribution of a language without attention to their meaning, our model will learn as long as our input data is given “plausible sequences”, meaning sequences that are really used in the target language regardless of meaning. In that regard, if we can augment a source sentence by replacing a word with a synonym or antonym, we are actually adding valuable information. Since the pre-training datasets that are usually used are from Wikipedia and news articles, there is a very high probability that the data contains numerous sentences where the difference between them is a single word.

Since we’re trying to augment the pre-training dataset of a language, the main issue is how we’re going to augment it. Since it is small, we won’t be able to have accurate word embeddings that allow us to make good word replacements, and since we try to use this for low resource languages, there’s most probably no digital dictionary like WordNet we can use. Thus we need to find a new way to reach our goal that doesn’t need any prior work and that can be used on any target language out of the box.

As the global idea is to augment the pre-training dataset, we came up different ideas on how to do that. In the next sections, we’ll be diving into the different methods we came up with to achieve the goal.

2 Heuristics-based methods (HBMs)

HBMs are methods that are based on logic derived from humans. In our context, HBMs will be ways to augment the data with simple handcrafted rules that do not depend on Machine Learning.

2.1 Easy Data Augmentation (EDA)

Published in 2015, the EDA paper[49] (that we already mentioned in the review chapter) is a set of 4 fast data augmentation techniques that are easy to implement and have shown improvements on NLP classification tasks. It especially performed well on very low data (less than 500 lines). The 4 techniques are:

- **Synonym Replacement:** Randomly replace words from the sentence with a synonym.
- **Random Insertion:** Insert a synonym of a word in the sentence in a random position.
- **Random Swap:** Swap two randomly chosen words in the sentence.
- **Random Deletion:** randomly remove a word in the sentence.

EDA uses the English WordNet [11] graph in two of its 4 techniques (replacement and insertion). Since we are working on languages that don't have a WordNet equivalent, we will try 2 variations of EDA: EDA without WordNet (only swapping and deleting) and EDA combined with another method (Graph or ML model) to replace WordNet.

2.2 N-gram Co-occurrence matrix (NCM)

We really liked the idea of Word-Word Co-Occurrence Matrix from the Glove [13] paper, which is a matrix that calculates the probability of each word in the vocabulary knowing the previous word, also known as 2-gram co-occurrence matrix, with 2-gram simply meaning 2 words. This idea was abandoned when we got into Deep Learning because two words aren't enough to get the overall context of a sentence. We however find this interesting, because in natural language there are multiple occasions of exact similar sequences that only differ in one word. We therefore have the concept of extending the idea to N-grams, with N being bigger than two. This would greatly shrink the size of the matrix, but give in contrast more valuable information.

To create an NCM, we first choose an integer N (the bigger N, the more context we can get, but this depends on the source dataset). Then we would create a matrix where the rows are the unique N-grams, the columns the next words for each N-gram in the whole dataset, and the values are the counts of the i-th word occurring after the j-th N-gram in the whole dataset. Table III.1 illustrates an example of NCM on an imaginary dataset with N=4 (the values are therefore fake and just for demonstration purpose).

We can then augment our dataset by separating each sentence into N+1-grams, and replacing the last word with the top X words from the built NCM.

Tableau III.1 – Example of an 4-Gram CM

	you	work	cat	yes	our	definition	...	games
I hope to see	9	1	1	0	3	1	...	2
I wish you a	0	0	1	0	0	0	...	0
As long as we	0	3	0	0	0	0	...	0
The definition of that	0	1	1	0	0	1	...	0
...	...	...	...	...	...	...	...	...
Thanks for your hard	0	5	0	0	0	0	...	1

3 Model-based Methods (MBMs)

MBMs are methods that are based on the inference of trained machine learning models. In our context, we believe that even if language models have been trained on small datasets, they should have learned at least a rough representation of the input. We can use that to our benefit if we know how to exploit it.

3.1 Contextual Augmentation using Word Embeddings (CAWE)

A Language Model learns a representation of a language, called an Embedding space. This embedding space is an N-dimensional Vector space where similar words are clustered together. In Figure, we can see an example of an N-dimensional embedding space from a trained language model that's flattened to 2D for better visibility.

[2]

As seen in figure III.1, words used in similar contexts are grouped together (verbs, synonyms, nouns...). Finding the most similar word to a queried one is done by calculating the difference between the query word vector and all other word vectors, also called vector distance, and returning the word with the smallest distance. CAWE is therefore the act of replacing a word in a sentence by the most similar word in the embedding space of the trained Language Model. CAWE takes into account context, meaning the previous and next words in the sentence, to give

[2]Example embedding space, https://www.researchgate.net/figure/2-dimensional-projection-of-the-word-embedding-space-with-a-zoom-on-the-word-prism-and_fig2_332088506s

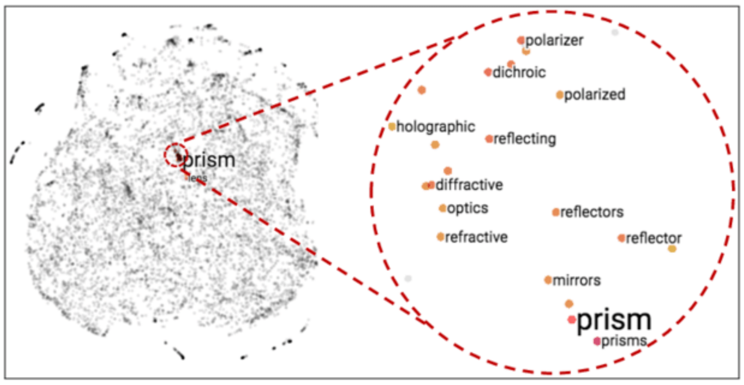


Figure III.1 – Example of an embedding space[1]

back the most adequate word. This is what Kobayashi, Kumar and Xie [43, 44, 46], mentioned in related works, did in their papers. We will use CAWE to augment the pre-training dataset itself.

3.2 Pre-training Data Augmentation (PDA)

We propose a general, model-agnostic, data-agnostic augmentation methodology for pre-training datasets using Language Models called Pre-training Data Augmentation (PDA). Illustrated in Figure III.2, this is how it works:

1. We train an initial Language model LM1 on an initial dataset D1. Even though it's not much, LM1 should still have learned a rough representation of the language from D1.

2. We use the trained language model LM1 to generate new sentences from D1, resulting in a bigger dataset D2.

3. We Train a new Language model LM2 on the new dataset D2.

Since even generating one example per sentence would result in doubling D1, The newly trained language model LM2 should be able to get better results on downstream tasks than the initial LM1. This is what we will try to prove in the experiments section.
While we already established that the "Data Augmentation" is going to be achieved using CAWE, there are still multiple other ways to do it. To be more specific, the difference is in what word to replace in each sentence. In the next section we'll discuss the different variations of choosing the words to replace.

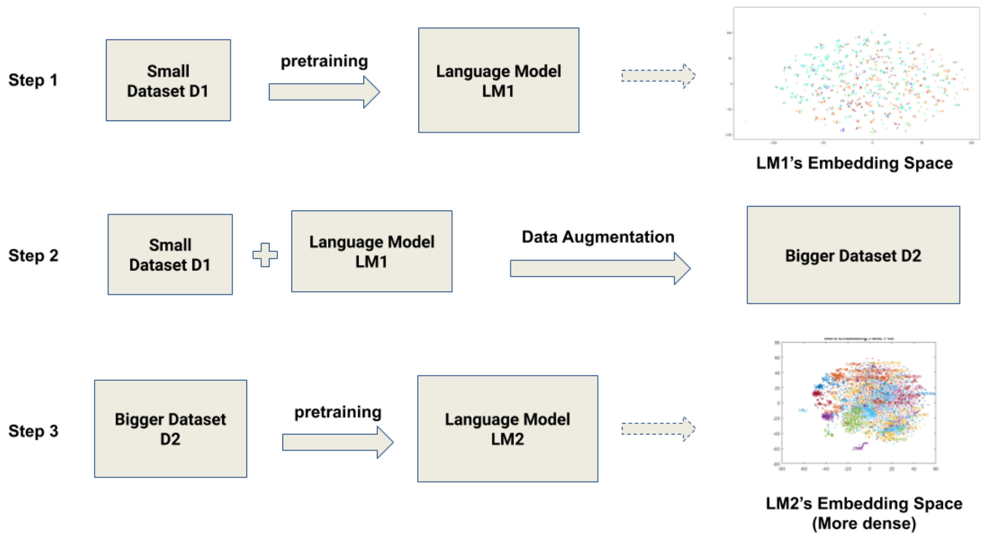


Figure III.2 – Our CAWE-PD pipeline

3.3 Choosing the words to replace

As a sentence contains multiple words, and each word can be replaced by N words using CAWE, the question of what words to replace arises.
With CAWE, there are two types of techniques:

- **Swapping:** replace target words with the most suitable.
- **Insertion:** insert new words in a sentence depending on its context.

There are also numerous ways to choose which words to apply the previous techniques on:

- **Random:** For each sentence, randomly choose. This may however favor a position too much.
- **Kfold:** For each sentence, choose all words. In case of swapping, swap all words once. In case of insertion, insert a new word in all spaces between tokens. Doing this for a full dataset would result in a very big dataset which will, in turn, need a lot of computation power to train the model on.
- **All In:** based on the Kfold cross validation[55] principle, we start with the first word of a sequence, then slide the position by 1 for each new sentence until we reach a sentence that's shorter than our current position. We then reset our position to the first word and repeat. Sorting the dataset beforehand helps a lot. We believe that this solution presents the best trade-off between completeness and performance.

There is also the option of doing this with two or more words in the same sentence.

4 Mixed methods (MMs)

MMs take the best of both worlds by combining parts of each.
In our case, an interesting MM is to combine EDA with a language model since Language Models are trained to cluster similar words together and to predict words given a context. We can therefore use a trained language model instead of WordNet for the synonym replacement and Insertion methods.

Conclusion

In this chapter we proposed a number of methods and techniques to augment pre-training datasets. Since this is an unexplored terrain we do not know yet how well these methods will fare and will therefore empirically explore their effectiveness in the next chapter.

Chapter IV

Experimentation

In this chapter, we will present the results of our experimentations and validate our theory.

1 Setup

Due to the amount of variables we have, it is important to first define what we have and don't have, as well as what we will choose to experiment on in the available time frame. Pre-training language models is time-consuming. BERT [26], one of the first LMs, takes 2 weeks to fully pre-train on Cloud V2-TPUs on, which are a lot faster than our hardware.

1.1 Hardware

We are using 1 Ubuntu machine with a GTX 1060 Super Graphics card with 6GB Dedicated GPU RAM. It's considered a high middle tier machine, so we will have to limit the scope of our experimentations.

1.2 Model

Since our hardware is limited, we will use ELECTRA [37], published in 2020, as our LM for pre-training, fine-tuning and augmenting. This is because it managed to get outstanding results, way better than the original BERT, for a fraction of the training time needed. This is very interesting given our hardware limitation and is the reason why we will go with it. We will also use the Small version of the ELECTRA model.

1.3 Language

While we would like to try our methods directly on low resource languages, the English language has a lot of already available benchmarks, codes, datasets and resources. We will therefore validate our ideas on the English language and in future works test on low resource languages and dialects.

1.4 Chosen augmentations

Since trying all augmentation methods will be too computationally expensive, we will only try a handful of them. Namely, we will only use 4 augmentation techniques for each base dataset:

- **EDA without WordNet:** For each sentence, we randomly delete and swap words. We will refer to this augmentation as just EDA.
- **EDA with WordNet:** For each sentence, we do EDA plus random insertion and replacement of words using the WordNet Graph. We will refer to this augmentation as EDA WordNet.
- **ELECTRA k-fold:** For each sentence, we choose a word that we mask and an ELECTRA model trained on the base dataset will generate 5 suggestions of words that fit the context. We will refer to this augmentation as ELECTRA K-fold.
- **EDA with ELECTRA:** For each sentence, we do EDA plus random insertion and replacement of words using an ELECTRA model trained on the base dataset. We will refer to this augmentation as EDA ELECTRA.

We will therefore have, for each base dataset, a total of 5 pre-training datasets (base included), and 5 pre-trained models (base model included). Each model will then further be fine-tuned on downstream tasks.

2 Datasets

In this section we will describe the datasets we will use throughout the experimentations.

2.1 Wikipedia Sentences

Wikipedia Sentences is an 823M text dataset of sentences taken from the English Wikipedia. Of it, we will randomly sample 100M. The resulting dataset's distribution of text by length is plotted in the figure. Since Wikipedia is the standard source for pre-training datasets for Transformers, we can then make a fair comparison with SOTA models.

2.2 Wikitext-2

The Wikitext Long Term Dependency Language Modeling Dataset [53] is a 4M text dataset used to benchmark models on the language learning task. It contains long paragraphs from

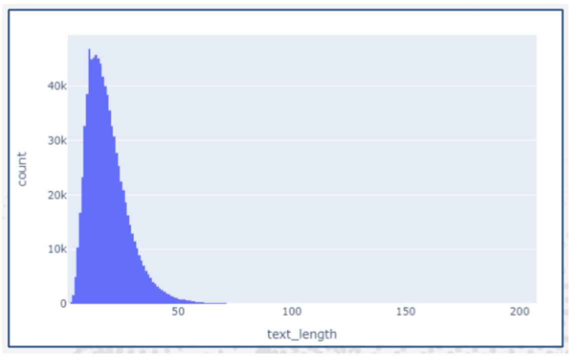


Figure IV.1 – Distribution of sentences by length in the Wikipedia dataset

Wikipedia about multiple subjects. We chose it because we want to see how well our ideas work on very small datasets.

2.3 Reddit Comments

Reddit Comments is a 2G text dataset of comments posed on the Reddit forum. Of it, we will also randomly sample 100M. The resulting dataset's distribution of text by length is plotted in the figure. We chose it because unlike the previous two, this dataset has informal text in the context of replies. Getting good results on this dataset too would validate that our model can work on any dataset.

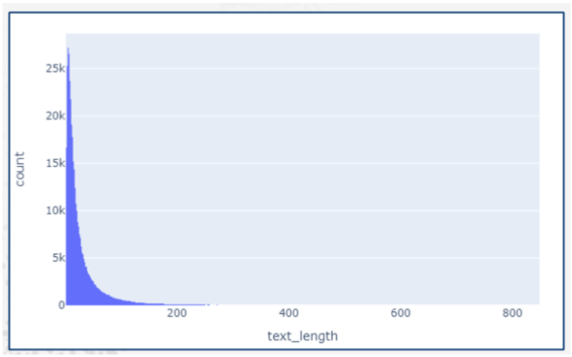


Figure IV.2 – Distribution of sentences by length in the Reddit dataset

3 Pre-training results

In this section, we discuss the results of the pre-training phase. For each base dataset, we will record the evaluation loss of the models trained on it. There are 5 models per dataset (base, EDA, EDA WordNet, EDA ELECTRA, ELECTRA Kfold). We will also evaluate models trained on a base dataset on the test set of the other datasets, to see how well these models and our augmentation techniques fare on never before seen data of different type and quality.

Since each training epoch takes between 2.5 hours and 10 hours on our hardware, we trained each model on a maximum of 10 epochs with Early Stopping (stops the training if the model is overfitting). We use an adaptive learning rate with an initial value of 2e-5 and a batch size of 16.

In the following, we will record the test losses of the models on the test sets of all our 3 datasets. **The lower the value the better.** To get a better understanding of the results, we will put both a numerical and a graphical representation.

To help with visibility, good results will be colored in green while bad results will be colored in red. Due to some unfortunate technical problems, some values are missing. They will be replaced with "N/A".

3.1 Results using the Wikipedia dataset

This section will explore the results of the Electra Small model trained on the Wikipedia dataset.

In Table IV.1, we can see that most of the augmentations on the Wikipedia dataset actually gave worse results than the base one. EDA was the only one successful with a 10% decrease in the loss. However, what's interesting is that all the augmentations gave us better results on the Reddit and Wikitext-2 test sets, even though Wikitext-2 is very similar to Wikipedia, while Reddit is very different.

Tableau IV.1 – Eval losses of models trained on Wikipedia

Model Name	Wikipedia Test	Reddit Test	Wikitext-2 Test
Base	10.8	33.27	33.5
EDA	**9.966 (-0.83)**	N/A	N/A
EDA WordNet	11.34 (+0.54)	28.17 (-5.10)	27.35 (-6.15)
EDA ELECTRA	23.99 (+13.19)	25.80 (-7.47)	25.22 (-8.28)
ELECTRA Kfold	14.62 (+3.82)	**24.69 (-8.58)**	**23.62 (-9.88)**

Figure IV.3 shows us more clearly the fluctuations of the losses per dataset and per augmentation. As we can see, for the Wikipedia Test set, the EDA augmentation gave us the best results while for both the Reddit and Wikitext-2 test sets, it was the ELECTRA Kfold augmentation that gave the best results.

While there wasn't an augmentation technique that always gave better results, the data has shown us that the best results on each dataset (represented in bold in the table) are from

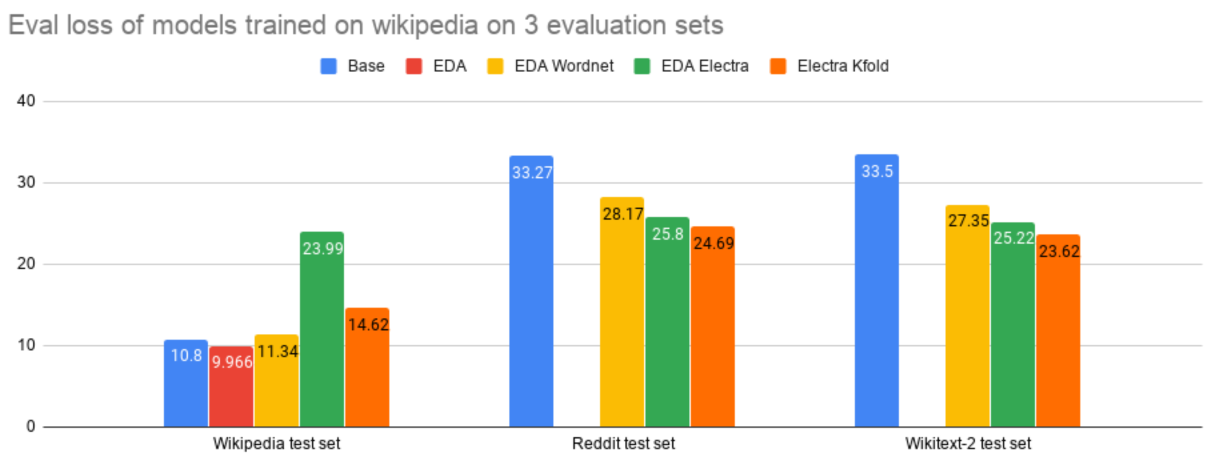


Figure IV.3 – Eval losses of models trained on Wikipedia

models pre-trained on augmented datasets.

3.2 Results using the Wikitext-2 dataset

This section will explore the results of the Electra Small model trained on the Wikitext-2 dataset.

In table IV.2, we can see that all our augmentations gave us better results on all test sets.

Tableau IV.2 – Eval losses of models trained on Wikitext-2

Model Name	Wikipedia Test	Reddit Test	Wikitext-2 Test
Base	31.16	32.40	32.5
EDA	28.26 (-2.9)	27.81 (-4.59)	17.64 (-14.64)
EDA WordNet	27.10 (-4.06)	26.41 (-5.99)	19.27 (-13.23)
EDA ELECTRA	27.75 (-3.41)	27.75 (-4.65)	15.13 (-17.37)
ELECTRA Kfold	27.77 (-3.39)	29.49 (-2.91)	22.13 (-10.37)

Figure IV.4 shows us more clearly the fluctuations of the losses per dataset and per augmentation. As we can see, there is a decrease in the losses of all models of all datasets on all test sets. The results on the Wikitext-2 Test set especially, were very pleasing as the losses nearly got divided by 2. This is very reassuring as it means the model is getting better in the use case we need it the most.

While there also wasn't an augmentation technique that always gave better results, We are very glad to notice that the best augmentation technique on the base dataset's test set was the EDA Electra one, because this technique does not need a previously built graph and thus can

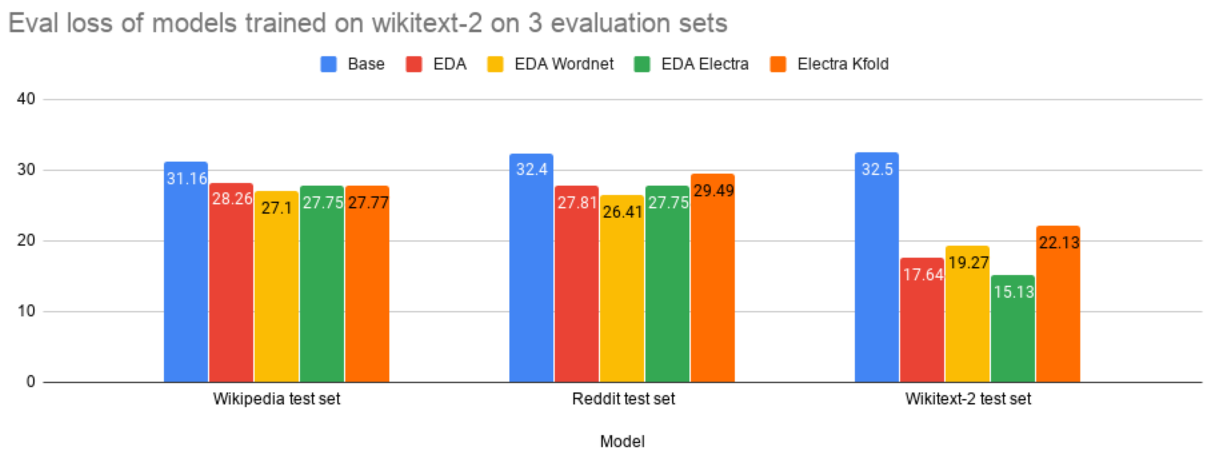


Figure IV.4 – Eval losses of models trained on Wikitext-2

be used on any data. One interesting remark would also be how WordNet actually gave worse results than plain EDA even though Wikitext-2 is also in English. By further investigating, we suspect that it is because the Wikitext-2 dataset was prepared years after WordNet and thus contained words it did not know, which reinforces our belief that we need a method that doesn't need maintenance.

3.3 Results using the Reddit dataset

This section will explore the results of the Electra Small model trained on the Reddit dataset.

DISCLAIMER: In the middle of our experimentations, our initially sampled Reddit dataset has been deleted. Since re-sampling will not give us the same dataset, we would have to redo all experimentations (a total of 250 hours for just the model training). Due to time constraints, we could not do that and will instead put the results we already have since they do hold valuable information.

In table IV.3, we can see that for the recorded results, we got mostly positive results. It still isn't understood why EDA failed badly on the Reddit Dataset.

Tableau IV.3 – Eval losses of models trained on Reddit

Model Name	Wikipedia Test	Reddit Test	Wikitext-2 Test
Base	44.84	9.45	44.51
EDA	34.25 (-10.59)	15.58 (+6.13)	31.97 (-12.54)
ELECTRA Kfold	N/A	9.245 (-0.20)	43.8 (-0.71)

Figure IV.5 shows us more clearly the fluctuations of the losses per dataset and per augmentation. As we can see, the increase and decrease of the losses are seemingly random.

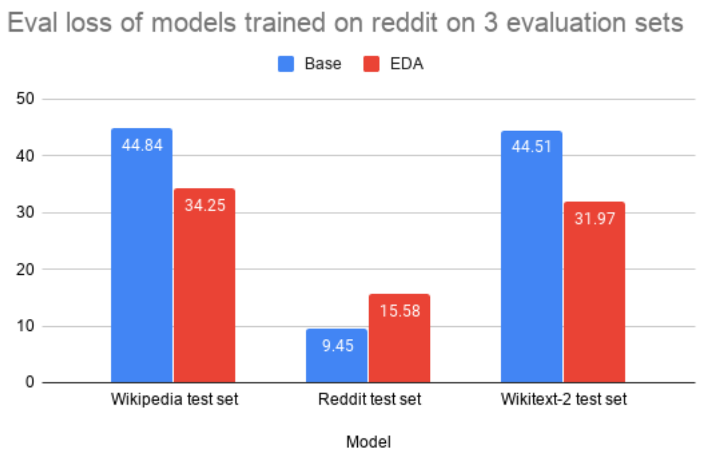


Figure IV.5 – Eval losses of models trained on Reddit

Since we don't have the complete results, we can't generalize in this example. However, the results still act according to our theory and most of the augmented datasets gave better results even if by a small margin.

3.4 Analysis and discussion

In this section we will analyze the overall results we obtained from the experimentations. Here is what we found out:

- We had a total of 36 experimentations: 9 base results and 27 augmentations.
- Only 4 experiments gave us negative results, meaning we have a success rate of 85
- All augmentation techniques had only one negative result. They are all equally successful.
- The Reddit Dataset has 1 negative result (EDA) while the Wikipedia Dataset has 3 (All except EDA)
- The Wikitext-2 had no negative results.
- Some experiments gave worse results on their own datasets but better on others'.
- For Wikipedia, only EDA gave good results.
- For Reddit, only EDA gave bad results.

While we can't make conclusions because of the randomness of the data, there is one thing we did remark. The Wikitext-2 models, who had a very sharp decrease in loss on all augmentation techniques when compared to Wikipedia and Reddit, is only 4M big, while both others are 100M. The EDA [49] paper shows that the smaller the datasets, the better the results, and it seems that we further proved that. This is fantastic news for endangered and very low resource languages. While it's true that the test losses in themselves aren't that great when compared to the eval losses of other models on Wikitext-2 and other benchmarks, we are operating on the basis that researchers working on very low resources languages have to do with what data they have and manage to get. We also didn't do any pre-processing and quality assurance as it is not our focus, meaning that it is possible to get even better results if data pre-processing is introduced in future works.

We can conclude that the pre-training phase was relatively successful, but that is not our end goal. In the next section, we will tackle the fine-tuning process on downstream tasks, which are the real life use cases we will need trained models on.

4 Fine-tuning results

In this section, we discuss the results of the fine-tuning phase. We will fine tune each of the previous 15 models on downstream tasks. As a reminder, a downstream task is a well-defined, very specific task like text classification, summarization, similarity detection etc. Each task has its own labeled dataset and its own evaluation metric. In our case, we will fine-tune our models on the GLUE Benchmark, a currently very popular collection of NLP tasks that the NLP community is using to benchmark their Transformers based models. In the next sections, we will introduce and explore GLUE and report the results of fine-tuning our models on it.

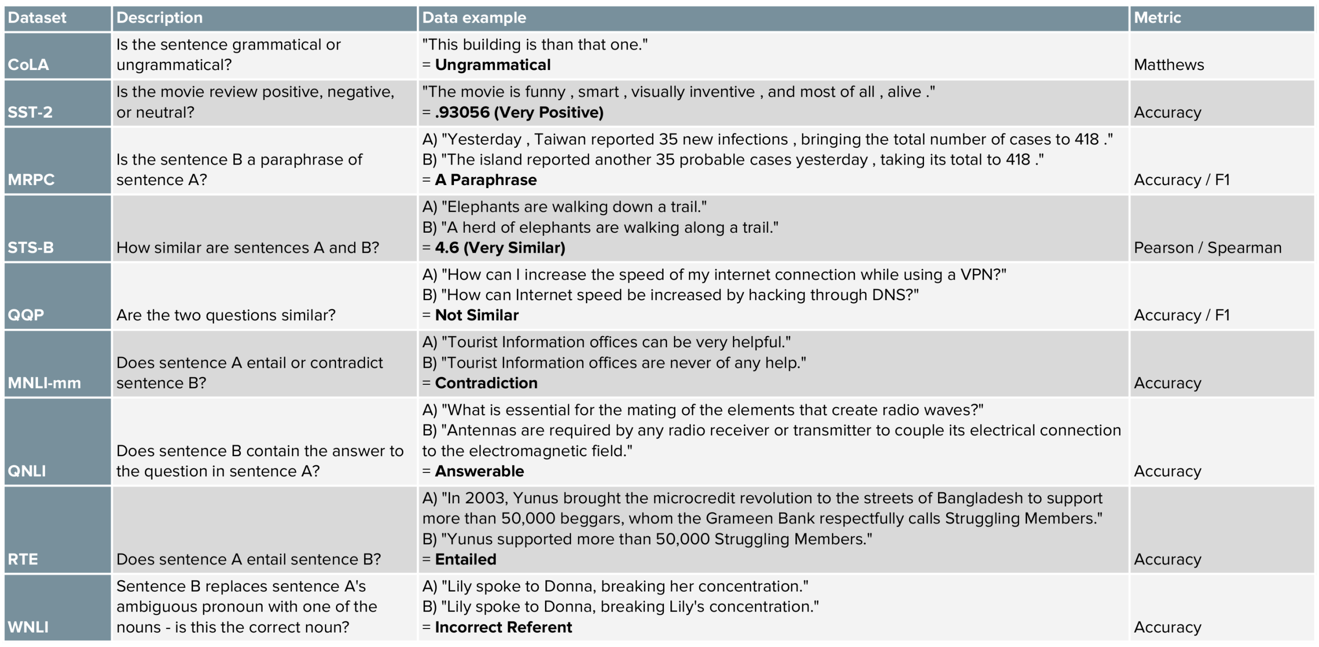
4.1 The GLUE Benchmark

The General Language Understanding Evaluation benchmark (GLUE) [29] is a set of NLP tasks with their corresponding labeled datasets that became the standard benchmarking method to compare performance of different models. The nine tasks are very diverse and test the models on a variety of different aspects of natural language. Figure IV.6 gives us a nice summary of the tasks and metrics.

[2]

We will train each model for 10 epochs on each task, this will result in a total of 25 hours of fine tuning per model. We will therefore get 10 values per model, we will also average all

[2]Glue Tasks, https://mccormickml.com/2019/11/05/GLUE/



Dataset	Description	Data example	Metric
CoLA	Is the sentence grammatical or ungrammatical?	"This building is than that one." = **Ungrammatical**	Matthews
SST-2	Is the movie review positive, negative, or neutral?	"The movie is funny , smart , visually inventive , and most of all , alive ." = **.93056 (Very Positive)**	Accuracy
MRPC	Is the sentence B a paraphrase of sentence A?	A) "Yesterday , Taiwan reported 35 new infections , bringing the total number of cases to 418 ." B) "The island reported another 35 probable cases yesterday , taking its total to 418 ." = **A Paraphrase**	Accuracy / F1
STS-B	How similar are sentences A and B?	A) "Elephants are walking down a trail." B) "A herd of elephants are walking along a trail." = **4.6 (Very Similar)**	Pearson / Spearman
QQP	Are the two questions similar?	A) "How can I increase the speed of my internet connection while using a VPN?" B) "How can Internet speed be increased by hacking through DNS?" = **Not Similar**	Accuracy / F1
MNLI-mm	Does sentence A entail or contradict sentence B?	A) "Tourist Information offices can be very helpful." B) "Tourist Information offices are never of any help." = **Contradiction**	Accuracy
QNLI	Does sentence B contain the answer to the question in sentence A?	A) "What is essential for the mating of the elements that create radio waves?" B) "Antennas are required by any radio receiver or transmitter to couple its electrical connection to the electromagnetic field." = **Answerable**	Accuracy
RTE	Does sentence A entail sentence B?	A) "In 2003, Yunus brought the microcredit revolution to the streets of Bangladesh to support more than 50,000 beggars, whom the Grameen Bank respectfully calls Struggling Members." B) "Yunus supported more than 50,000 Struggling Members." = **Entailed**	Accuracy
WNLI	Sentence B replaces sentence A's ambiguous pronoun with one of the nouns - is this the correct noun?	A) "Lily spoke to Donna, breaking her concentration." B) "Lily spoke to Donna, breaking Lily's concentration." = **Incorrect Referent**	Accuracy

Figure IV.6 – All GLUE tasks[1]

of them in a new value called G-score, which is the general score of the model. This will be used to globally evaluate the models. For all evaluation metrics, **the higher the value the better.**

4.2 Results on the Wikipedia dataset

This section will explore the results of the fine-tuning of the ELECTRA Small models trained on the Wikipedia dataset on the GLUE Benchmark. For all the evaluations, the higher the better.

Tableau IV.4 – Fine-tune results of models trained on Wikipedia on the Glue Benchmark

Model	CoLA	RTE	SST-2	STS-B	WNLI	MRPC	QNLI	QQP	MNLI	G-score
Base	7.5	53.1	77.4	20.62	49.2	78.7 68.6	60.5	82.0 77.7	65.73 67.31	54.29
EDA	11.49 (+3.99)	51.62 (-1.48)	82 (+4.8)	17.1 (-3.52)	56.34 (+7.14)	78.83 (+0.13) 69.85 (+1.25)	58.14 (-2.36)	80.58 (-1.42) 75.68 (-2.02)	62.64 (-3.09) 63.27 (-4.04)	54.68 (+0.39)
EDA Wordnet	0 (-7.5)	50.9 (-2.2)	80.4 (+3)	76.3 (+55.68)	53.5 (+4.3)	83 (+4.3) 73 (+4.4)	78.6 (+18.1)	84.85 (+2.85) 80.14 (+2.44)	68.31 (+2.58) 68.55 (+1.24)	63.18 (+8.89)
EDA Electra	0 (-7.5)	53.79 (+0.69)	81.65 +(4.25)	19 (-1.62)	56.34 (+7.14)	76.31 (-2.39) 64.71 (-3.89)	71.99 (+11.49)	81.6 (-0.4) 76.79 (-0.91)	62.28 (-3.45) 63.80 (-3.51)	55.06 (+0.77)
Electra kfold	7.6 (+0.1)	51.6 (-1.5)	79.9 (+2.5)	64.19 (+43.57)	49.3 (+0.1)	80.5 (+1.8) 69.9 (+1.3)	75.7 (+15.2)	84.02 (+2.02) 79.42 (+1.72)	65.83 (+0.1) 66.79 (-0.52)	61.78 (+7.49)

In table IV.4, we can see that the models trained on augmented datasets did better than the base model on some of the tasks. 23 of the 36 tasks had positive results, meaning that we have a success rate of 64%. We can also see that while each augmented model has its own ups and downs, we always get better results on the SST-2 and WNLI tasks and although only half the STS-B results are positive, they are far above the base results. When we look back at the GLUE tasks in figure, we can see that WNLI and STS-B are tasks that operate on the semantic similarity of sentences with different words, which is exactly how we augmented our datasets in the first place. The SST-2 task however is a sentiment analysis classification task. While it's not directly related to semantic similarity, it is however understandable since our augmentation techniques replaces words with similar ones in given contexts, so it may have generated other ways to express sentiments. Another interesting thing to point out is that, albeit not in the same model, augmentation improved results in all tasks. This means that regardless of the task, there is always a way to get better results through augmentation. Finally, the G-scores in the table (average of all metrics) show us that augmented models are in general better models than the base ones.

4.3 results on the Wikitext-2 dataset

This section will explore the results of the fine-tuning of the ELECTRA Small models trained on the Wikitext-2 dataset on the GLUE Benchmark. For all the evaluations, the higher the better.

Tableau IV.5 – Fine-tune results of models trained on Wikitext-2 on the Glue Benchmark

Model	CoLA	RTE	SST-2	STS-B	WNLI	MRPC	QNLI	QQP	MNLI	G-score
Base	-0.82	47.29	80.6	42.54	45.07	76.95 66.66	71.11	83.82 78.61	66.82 68.14	56.25
EDA	0 (+0.82)	55.23 (+10.82)	80.39 (-0.21)	10.69 (-31.85)	59.15 (+14.08)	80.56 (+3.61) 69.61 (+2.95)	73.88 (+2.77)	82.15 (-1.67) 76.97 (-1.64)	62.72 (-4.1) 63.72 (-4.42)	55.25 (-1)
EDA WordNet	0 (+0.82)	53.43 (+6.14)	81.77 (+1.17)	15.79 (-26.75)	53.52 (+8.45)	81.12 (+4.24) 69.12 (+2.46)	73.55 (+2.44)	80.88 (-2.94) 76.42 (-2.19)	64.79 (-2.07) 65.77 (-2.37)	55.24 (-1.01)
EDA Electra	4.1 (+4.92)	47.29 (0)	80.6 (0)	70.86 (+28.32)	54.9 (+9.83)	77.5 (+0.55) 67.4 (+0.75)	77.61 (+6.5)	83.9 (+0.08) 79.5 (+0.89)	67.71 (+0.89) 68.63 (+0.49)	61.96 (+5.71)
Electra Kfold	2.88 (+3.7)	45.49 (-1.8)	78.44 (-2.16)	44.01 (+1.47)	56.34 (+11.27)	81.12 (+4.17) 70.34 (+3.68)	74.44 (+3.33)	82.3 (-1.52) 77.41 (-1.2)	65.85 (-0.97) 65.95 (-2.19)	58.12 (+1.87)

In table IV.5, we can see that, similarly to the models trained on Wikipedia, these models also did better on some of the tasks. 25 of the 36 tasks had positive results, which means that we have a success rate of 69%. This time, unlike Wikipedia, our model does not always get better results on SST-2, but instead always had better results in 4 tasks: CoLA, WNLI, MRPC and QNLI. We already know why WNLI would be better, and upon verification, we can see that MRPC also has a sense of similarity as it treats paraphrasing. Besides, even in Wikipedia we had better results in MRPC in all but one case. CoLA being a test of grammatical correctness, we honestly do not know how randomly swapping and deleting words through EDA, and replacing a word by 5 suggestions that may not be good fits, managed to get us a better score in this task. We also do not know yet why QNLI benefitted from our augmentations. We are however, very pleased by the results. One particularly interesting result this time is the EDA Electra model, as we can see, it is the first time we get a perfect model, meaning all the results are equal or better than the base model. Finally, the GLUE scores show us that only 2 of the augmentations are better general models, but that is mostly because of the STS-B's sharp decrease. The rest of the tasks' results aren't far from the base models'.

4.4 results on the Reddit dataset

This section will explore the results of the fine-tuning of the ELECTRA Small models trained on the Reddit dataset on the GLUE Benchmark. For all the evaluations, the higher the better. Since we only have results for 2 models, we will discuss without jumping to conclusions.

Tableau IV.6 – Fine-tune results of models trained on Reddit on the Glue Benchmark

Model	CoLA	RTE	SST-2	STS-B	WNLI	MRPC	QNLI	QQP	MNLI	G-score
Base	11.17	53.07	82.34	19.69	56.34	77.16 67.64	59.39	80.59 75.68	62.64 63.27	55.05
EDA	6.52 (-4.65)	57.76 (+4.69)	80.85 (-1.49)	73.36 (+53.67)	52.11 (-4.23)	81.56 (+4.4) 69.85 (+2.21)	77.19 (+17.8)	83.82 (+3.23) 78.73 (+3.05)	67.11 (+4.74) 67.47 (+4.2)	63.56 (+8.51)

In table IV.6, we can see that similar to the models trained on both Wikipedia and Wikitext-2, these models also did better on some of the tasks. 6 out of the 9 tasks had positive results, meaning that we have a success rate of 67%. This model, even though of a different nature than the previous datasets, seems to be following the same steps as Wikitext-2, as it had better results in the same tasks as it (WNLI, QNLI, QQP and RTE). The EDA augmented model's G-score climbed up 8.5%, which is a substantial increase. It is now the highest scoring model in our experimentations.

4.5 Analysis and discussion

4.5.1 Results analysis

In this section we will analyze the total results we obtained from our experiments. Here's what we found out:

- We had a total of 108 experimentations: 27 base results and 81 augmentations.
- 24 augmentations gave us negative results, which means that we have a success rate of 70%.
- In total, we have 12 fine-tuned models: 3 base and 9 augmented.
- 7 out of these augmented models were better overall models than the base ones, which means a success rate of 78%. All augmented models would have been better if not for the

immense drop in STS-B task, which means that in real life, we can use the augmented models for tasks other than the ones similar to STS-B, and the base models for the STS-B like task. In that case, our success rate would grow to 89%.

- There is no best augmentation method for a given task, it depends on the dataset.
- There are augmentations that never gave good results on certain tasks. For example, EDA never gave better results for QQP, MNLI and STS-B while ELECTRA Kfold never gave better results on RTE.
- The best augmentations gave the Wikipedia-based model a boost of 9% and the Wikitext-2 based model a boost of 6%. The only augmentation on the Reddit-based model gave it an 8.5% boost. These are substantial and satisfying ameliorations.

4.5.2 Which model is the best for each task?

We also want to see which models performed the best on each task. In table IV.7, we note the best model for each task from its metric. We do this twice: with and without WordNet. This is because WordNet is something exclusive to the English language while we want our work to be language-agnostic. For visibility issues, we will refer to Wikipedia as Wiki and Wikitext-2 as Wiki-2.

Tableau IV.7 – Best models for each GLUE task

Model	CoLA	RTE	SST-2	STS-B	WNLI	MRPC	QNLI	QQP	MNLI
Best model	Wiki EDA	Reddit EDA	Reddit Base	Wiki EDA WordNet	Wiki-2 EDA	Wiki EDA WordNet	Wiki EDA WordNet	Wiki EDA WordNet	Wiki EDA WordNet
Without WordNet	Wiki EDA	Reddit EDA	Reddit Base	Reddit EDA	Wiki-2 EDA	Reddit EDA	Wiki-2 EDA Electra	Reddit EDA	Wiki-2 EDA Electra

If we take into account WordNet:

- the EDA + WordNet augmentation scored the best on 5 out of the 9 tasks. For 3 of the other 4 tasks, a simple EDA-only augmentation was enough to get the best scores. The last task goes to a base model.
- The Wikipedia dataset's models scored the best in 6 out of the 9 tasks, with Reddit scoring best in 2 and Wiki-2 in just one.

If we don't take into account WordNet:

- The EDA-only augmentation scored best in 6 of the 9 tasks, with EDA + ELECTRA scoring best in 2 and a Base model getting 1.
- This time however, the Wikipedia Dataset models only scored best in 1 task, while the Wiki-2 ones scored best in 3 tasks and the Reddit ones in 5 tasks.

While we couldn't figure out an absolute best method, we do now know that augmented models usually fare better than base ones and thus are to be considered in one's work.

4.5.3 Can these models compare against models trained on big datasets?

At this point, we are also interested in knowing how well our models, trained on datasets between 4M and 600M (max augmented dataset), fare against bigger models trained on gigabytes of text. To do that, we will select the GLUE results of some models from the GLUE leaderboard and compare them to ours. More precisely, we will compare our best results with those of ELECTRA Small (the same model architecture as ours, but trained on 13G of text), BERT-BASE (medium-small Architecture trained on 13G of text) and GLUE human baselines. In this table we will not calculate G-scores because we are using the best scores from multiple models.

Tableau IV.8 – Comparison between our models, models trained on more data and human baselines

Model	CoLA	RTE	SST-2	STS-B	WNLI	MRPC	QNLI	QQP	MNLI
Human Baselines	66.4	93.6	97.8	92.7 92.6	95.9	86.3 80.8	91.2	80.4 59.5	92.0 92.8
BERT Base	52.1	66.4	93.5	87.1 85.8	65.1	88.9 84.8	90.5	89.2 71.2	84.6 83.4
ELECTRA Small	55.6	63.6	91.1	85.6 84.6	65.1	89 84.9	88.3	88.0 70.4	81.6 81.2
Our best results (all models)	11.49	57.76 *	82.34 **	76.3 ***	59.15 ****	83 73 ***	78.6 ***	84.85 80.14 ***	68.31 68.55 ***
Our best results (no WordNet)	11.49	57.76 *	82.34 **	73.6 *	59.15 ****	81.56 69.85 *	77.61 X	83.82 78.73 *	67.71 68.63 X

* Wikipedia EDA — ** Reddit EDA — *** Reddit base — **** Wikipedia EDA wordNet — ***** Wikitext-2 EDA [X Wikitext 2 EDA Electra

From table IV.8, we can see that:

- BERT and ELECTRA are very close to each other so we will just compare to ELECTRA.
- While our CoLA scores are way below the other models, we are pretty close in the others.

- In particular, RTE, WNLI, MRPC and QQP all are within 6 points of the ELECTRA-Small results, while these models have been trained on approximately 3.5‘% of the data.
- SST-2, STS-B and QNLI are all within 10 points of ELECTRA-Small while MNLI is within 13.
- Even if we remove the EDA WordNet models, we only lose 2-3 points per task.
- Our models beat the QQP Human Baselines, both with and without WordNet (further investigation show that all our 15 models beat the QQP Human Baseline).

Conclusion

From the experimentations we saw in this chapter, we can now confidently say that:

- Augmenting the pre-training datasets gives better models and better results in the downstream tasks even with very small data.
- Each downstream task has its own particularities and needs, which means that type of augmentation to choose will depend on the final use case.
- We don't need huge text datasets to get good models ready to be used.

  We present in the next chapter reflections we had while performing the experimentations as well as ideas to improve our work.

Chapter V

Reflections and Improvement Ideas

Since we tackled a rather unexplored terrain, we observed a number of interesting things worth sharing. We also had a lot of ideas to improve our results, both directly and indirectly. In this chapter will we be sharing these thoughts to lay the foundations for future works.

1 Direct improvement ideas

These ideas are directly related to the work presented in this thesis. They represent the continuation of the experiments since we couldn't try everything in the limited time we had.

1.1 Further pre-train the base model

In our experimentations, once we generate a new dataset using a model trained on a base dataset, we throw that model out and pre-train a new one from scratch. What if instead, we further pre-train that trained base model with the augmented dataset? Would it give us better results or would it overfit? It it was already overfitting, would it stop overfitting and start correctly learning? Would this even newer model be able to generate even better augmentations?

1.2 Preprocessing

Data preprocessing is the step of either removing unwanted portions of text that could harm our model's learning, modifying parts of it and adding missing parts. We did not do any of these in our work because our scope is to validate the idea of augmenting pre-training datasets. We however believe that preprocessing can boost the performance of the models. In the figures, we plot the pre-training learning curves for our models (we applied the log function to make it easier to see).

As we can see from figure V.1, the models trained on Wikipedia and Reddit (left side) kept overfitting (losses kept going up) and were interrupted by the Early Stopping mechanism while Wikitext-2 (right side) models kept learning without overfitting (losses kept going down). We believe it has to do with the data itself. While the Wikitext-2 dataset has been cautiously prepared and maintained through the years, the Wikipedia and Reddit ones were just random

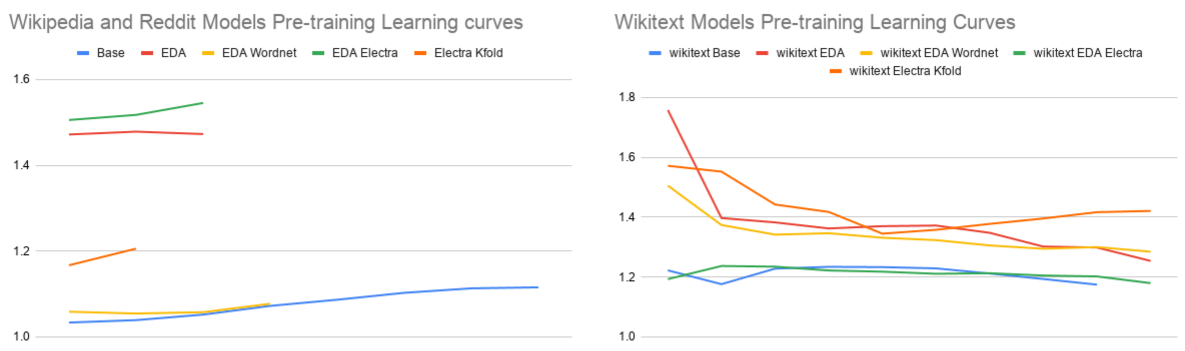


Figure V.1 – Pre-training learning curves: Wikipedia and Reddit on the left and Wikitext-2 on the right

samples from other preprocessed datasets. This means that with the proper preprocessing, our Wikipedia and Reddit models may have obtained even better results.

1.3 Augmentation techniques

In section 2.3, we introduced some augmentation techniques. Not only did we not manage to try them all, but there could be more ways that could be better than ours.

- Using another model than ELECTRA with EDA: ELECTRA-small's generator wasn't made to be very good at generation, just enough to train the discriminator.
- Multiple words augmentation in the same sentence: Replace or insert multiple words In the same sequence.
- Making a very small WordNet-like dictionary: While a whole graph like WordNet would take a lot of effort and time to make, one could, in 20 minutes, make a dictionary of common verbs, nouns, adverbs and synonyms in the desired language and use that.
- Using NCM: as discussed in section 2.2, NCM could generate very high quality sentences as it only takes into account what's existing in the source dataset.

1.4 Importance of the base dataset

While training our base models, we stumbled upon interesting findings. Table shows us the fine-tuning results of the base models only. Colored in green are the best results.

Tableau V.1 – Base datasets fine-tuning losses

Dataset	CoLA	RTE	SST-2	STS-B	WNLI	MRPC	QNLI	QQP	MNLI	G-score
Reddit	**11.17**	53.07	**82.34**	19.69	**56.34**	77.16 67.64	59.39	80.59 75.68	62.64 63.27	55.05
Wikitext-2	-0.82	47.29	80.6	**42.54**	45.07	76.95 66.66	**71.11**	**83.82** **78.61**	**66.82** **68.14**	**56.25**
Wikipedia	7.5	**53.1**	77.4	20.62	49.2	**78.7** **68.6**	60.5	82.0 77.7	65.73 67.31	54.29

We can see from table V.4 that by just changing the base dataset, we can get different results for the same tasks. For the CoLA task for example, Reddit made a huge leap when compared to the others. It is understandable since Reddit comments are very prone to be ungrammatical and informal, thus unintentionally helping in the task. For STS-B, Wikitext-2's score is more than the double of the others'. It is thus important, when choosing what type of data to collect, to think of the task we want it to be good at. Following this, in table V.2 we find the pre-training results of the base models. As we can see, there is nothing out of the ordinary, the models always perform better on their own data and quite badly on others.

Tableau V.2 – Base models pre-training results

Model Name	Wikipedia Test	Reddit Test	Wikitext-2 Test
Wikipedia Base	10.8	33.27	33.5
Reddit Base	44.84	9.45	44.51
Wikitext-2 Base	31.16	32.4	32.5

We had just enough time left for a final experiment, so we re-sampled a new base dataset from the Reddit dataset and pre-trained then fine-tuned a new base Reddit model. The following table contains the pre-training results of the the new Reddit model, called Reddit-2, and the old Reddit model.

Tableau V.3 – Reddit vs Reddit-2 pre-training results comparison

Model Name	Wikipedia Test	Reddit Test	Wikitext-2 Test
Reddit Base	44.84	9.45	44.51
Reddit-2 Base	30.85	29.21	30

As we can see in table V.3, the new Reddit model does a lot worse than the previous one on its own data, but a lot better on the others. This means that whether our model does well or not depends on how the initial data got sampled and we have been lucky to get an initial sample that got us a relatively good result. When pre-training and fine-tuning takes at least 50 hours on a medium-high GPU, one can't just sample multiple times and hope for the best.

We will explore this point in the next section. We also fine-tuned our new Reddit-2 model on GLUE. Table shows the results:

Tableau V.4 – Reddit vs Reddit-2 fine-tuning results comparison

Dataset	CoLA	RTE	SST-2	STS-B	WNLI	MRPC	QNLI	QQP	MNLI	G-score
Reddit	**11.17**	53.07	**82.34**	19.69	**56.34**	77.16/67.64	59.39	80.59/75.68	62.64/63.27	55.05
Reddit-2	0	51.99	81.57	9.54	46.48	80.54/67.89	60.88	78.86/69.42	60.29/60.54	51.02

The table shows us that while the new model fared badly when compared to the old model, it did still get better results on 2 tasks. This means that somehow, this dataset has elements that are well-suited for MRPC-like and QNLI-like tasks. To conclude, this section shows us that the base dataset play an important role in the final results. We are thus getting more into the idea of Quality of Data instead of Quantity of it. We will discuss this concept in the next section.

2 Indirect Improvement ideas

These are the ideas that are not directly related to our work in this thesis, but to the overall goal of better language learning with less data.

2.1 Quality of Data (QoD)

We briefly talked about Quality of Data, but this term holds more to it. We believe that QoD should be looked at more in-depth in the future. We will give some examples in this section.

2.1.1 Data usefulness index

From the previous tables and figures, one question kept repeating: how did the models trained on the Wikitext-2 dataset, that's only 4M big, get results that are better than models trained on 100M datasets like Reddit and Wikipedia, and results that are nearly comparable to models trained on 13G big datasets like ELECTRA small and BERT Base? We believe that the quality of the information held in the data can have a lot more influence than the quantity. The 13 Gigs of text used in bigger models come from sources like Wikipedia and news articles, in which content has similar structure, context and representations. Therefore, there has to be a good number of similar if not identical content that can be dropped without hurting performance. We believe that instead of going for quantity, we should find a way to calculate the quality of a dataset. An interesting future work perspective could be to find a way to somehow calculate how good a dataset is or what's the percentage of valuable information it holds, or how valuable is it. This can help get rid of useless content in an initial dataset and thus compress it to a smaller, denser and better dataset. Not only would this help get better results, it would also make training and augmenting faster. This can be further developed to calculate the usefulness of the dataset per downstream task and can become a standard procedure in any NLP work.

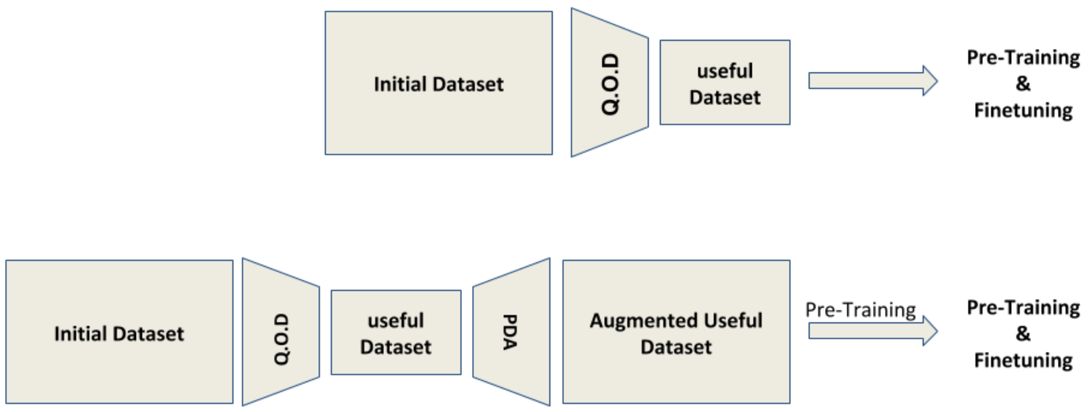


Figure V.2 – Quality of Data Procedure (top) and Quality of Data + pre-training Data Augmentation (bottom)

figure V.2 (top part) illustrates how, if we had a way to calculate the quality of a dataset, we could filter out the useless data and only work with the most useful information of a source dataset, thus gaining in training time while not losing performance. The bottom part shows how, if we combine the QoD with our PDA method, we could a big dataset of only useful data. One could even go further and apply QoD again on the augmented data to further filter it out.

2.1.2 What is my data lacking?

One way to explain our experiments results, is that some datasets contain information that others don't. Language has a lot of content: questions, sentences, paragraphs, symbols, meaning. Till now, we don't know exactly what our data holds and how much it does for each. So we fill the gap by getting more data. However, if we could find a way to know what our data lacks, this could get us further in terms of efficiency. An idea to do this is to somehow make a function that can take a dataset, train a model on it, compare its embedding space with embedding spaces of better models, and see where there are gaps. Figure V.3 better illustrated this.

If we have this, we can know where to focus our efforts and get just the right data to get better results.

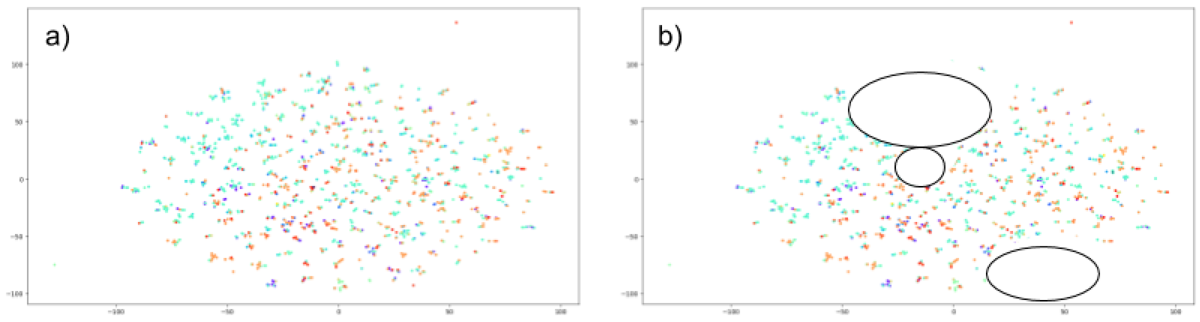


Figure V.3 – what we think our data is like (left) vs how it can really be (right)

2.2 Embedding space translation

In figure III.1, we showed the concept of embedding space which represents what a language model learns. Embedding spaces share similarities even on different languages. We could use the embedding spaces of other better models to correct our model's embedding space. To do so, we make a small list of words and their translations in the other models' languages, then a function would map the words to the vector space, and it would identify clusters of similar words, calculate vector distances and map those distances to our model's embedding space, thus correcting the clusters and the distances.

Conclusion

In this chapter we gave an in-depth analysis on how we can further improve our own work and more generally to attain the bigger goal of getting good results with less data.

General conclusion

In this master thesis, we were interested in seeing how well state-of-the-art language models, that normally require a huge amount of data, would fare on very small datasets and how we can make them learn better from them.

More specifically, we were interested in the idea of Data Augmentation and introduced a novel approach that focuses on augmenting the pre-training datasets. Within this same approach, we also proposed numerous methods on how to reach this goal.

We tested our idea on the English language because of the already available datasets, benchmarks, and codes. We tested on 3 different datasets that differ in size (from 4MB to 100MB), manner of speech (formal and informal) and nature (sentences, paragraphs and comments). For each, we pre-trained a type of Transformers based model called ELECTRA, then we fine-tuned them on the GLUE benchmark, consisting of 10 tasks.

We proved that augmenting the pre-training datasets really did improve results and while the amount of the improvement depends on the base dataset and the desired task, it generally gives better results.

We also presented a number of improvement ideas, both directly linked to our work and indirectly, of which 2 stand out: NCM and Quality of Data. The former is a simple statistic based data augmentation technique with a high potential to be extremely effective, and the latter an approach that focuses on getting the most of our available data instead of blindly looking for more. Both of these present very promising future works.

We also need to test our data augmentation approach on real low resource languages, to be sure of its effectiveness.

This is very important for us because we firmly believe that the democratization of technology can greatly benefit human civilization, and that letting it in the hands of the few will never let us explore its full potential. From making Machine Learning models need less computational power to optimal exploitation of available data, anything that can let people with standard equipment exploit state-of-the-art solutions can only make our world and our lives better.

Bibliography

[1] Ekin D Cubuk, Barret Zoph, Vijay Vasudevan, and Quoc V Le Google Brain. AutoAugment: Learning Augmentation Strategies from Data. Technical report. 3

[2] Akito Takeki, Daiki Ikami, and Kiyoharu Aizawa. Parallel Grid Pooling for Data Augmentation. Technical report. 3

[3] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random Erasing Data Augmentation. Technical report. 3

[4] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Data Augmentation using Random Image Cropping and Patching for Deep CNNs. Technical report. 3

[5] Daniel V Ruiz, Bruno A Krinski, and Eduardo Todt. ANDA: A Novel Data Augmentation Technique Applied to Salient Object Detection. Technical report. 3

[6] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. Technical report. 3

[7] T M Cover and P E Hart. Approximate formulas for the information transmitted bv a discrete communication channel. Technical Report 1 (1952). 4

[8] Corinna Cortes. Support-Vector Networks. Technical report (1995). 4

[9] D. E. Rumelhart and J. L. McClelland. *Learning Internal Representations by Error Propagation* pages 318–362. (1987). 4

[10] Karen Spärck Jones. *A statistical interpretation of term specificity and its application in retrieval.* Journal of Documentation **60**, 11–21 (2004). 6, 7

[11] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to WordNet: An On-line Lexical Database. Technical report. 6, 7, 23

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. Technical report. 8

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. Technical report. 9, 23

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. Technical report. 9

[15] Mikael Bodén. A guide to recurrent neural networks and backpropagation. (2001). 10

[16] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory*. Neural Computation **9**(8), 1735–1780 (1997). 11

[17] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Technical report. 11

[18] Samira Shabanian, Devansh Arpit, Adam Trischler, and Yoshua Bengio. Variational bi-lstms, (2017). 11

[19] Matthew E Peters, Mark Neumann, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. Technical report. 11

[20] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. Technical report. 12

[21] Lisa Torrey and Jude Shavlik. Transfer Learning. Technical report. 12

[22] Mahbub Hussain, Jordan J Bird, and Diego R Faria. A Study on CNN Transfer Learning for Image Classification. Technical report. 12

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. *Attention is all you need*. Advances in Neural Information Processing Systems **2017-Decem**(Nips), 5999–6009 (2017). 12, 13

[24] Alec Radford. Improving language understanding by generative pre-training. (2018). 13

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. (2019). 13, 15, 18

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. (Mlm) (2018). 13, 15, 18, 19, 28

[27] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. pages 1–18 (2019). 13, 15, 18, 19

[28] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, and Paul G Allen. RoBERTa: A Robustly Optimized BERT Pretraining Approach. Technical report (2019). 13, 18, 19

[29] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND-ING. Technical report. 14, 35

[30] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, (2019). 15

[31] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. *ERNIE: enhanced language representation with informative entities.* CoRR **abs/1905.07129** (2019). 15

[32] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding, (2019). 15

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu Google. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Technical report. 15, 18

[34] Jared Kaplan, Sam McCandlish, Tom Henighan OpenAI, Tom B Brown OpenAI, Benjamin Chess OpenAI, Rewon Child OpenAI, Scott Gray OpenAI, Alec Radford OpenAI, Jeffrey Wu OpenAI, and Dario Amodei OpenAI. Scaling Laws for Neural Language Models. Technical report (2020). 15, 18, 22

[35] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam Mccandlish, Alec Radford, Ilya Sutskever, and Dario Amodei Openai. Language Models are Few-Shot Learners. Technical report (2020). 15, 18

[36] Stephen R. Anderson. How many Languages are there in the World? Technical report (2010). 16

[37] Kevin Clark, Minh-Thang Luong, Google Brain, Quoc V Le Google Brain, and Christopher D Manning. ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS. Technical report. 19, 28

[38] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. Technical report. 19

[39] Geoffrey Hinton and Jeff Dean. Distilling the Knowledge in a Neural Network. Technical report (2015). 19

[40] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. Poor Man's BERT: Smaller and Faster Transformer Models. Technical report. 19

[41] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. Technical report. 19

[42] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8BERT: Quantized 8Bit BERT. Technical report (2019). 19

[43] Sosuke Kobayashi. Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations. Technical report. 19, 22, 25

[44] Varun Kumar, Alexa Ai, Ashutosh Choudhary, and Eunah Cho. (No Title). Technical report (2020). 19, 22, 25

[45] Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. Conditional BERT Contextual Augmentation. Technical report. 19

[46] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. UNSUPERVISED DATA AUGMENTATION FOR CONSISTENCY TRAINING. Technical report. 20, 22, 25

[47] Dongju Park and Chang Wook Ahn. *Self-supervised contextual data augmentation for natural language processing*. Symmetry **11**(11) (2019). 20

[48] Shujuan Yu, Jie Yang, Danlei Liu, Runqi Li, Yun Zhang, and Shengmei Zhao. *Hierarchical data augmentation and the application in text classification*. IEEE Access **7**, 185476–185485 (2019). 20

[49] Jason Wei and Kai Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. Technical report. 20, 23, 35

[50] Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-Yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Learning Semantic Textual Similarity from Conversations. Technical report. 20

[51] Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. Soft Contextual Data Augmentation for Neural Machine Translation. Technical report. 20

[52] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. Technical report. 20

[53] Stephen Merity, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. Technical report. 21, 29

[54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: A simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research **15**(56), 1929–1958 (2014). 21

[55] Payam Refaeilzadeh, Lei Tang, and Huan Liu. C Cross-Validation. Technical report. 26