Natural Language Processing, a Machine Learning approach

Diego Manuel Rodríguez Sánchez

Submitted in partial fulfilment of the requirements of Edinburgh Napier University for the Degree of Master of Science in Computing

School of Computing

August 2020

MSc dissertation check list

Milestones	Date of completion	Target deadline
Proposal		Week 3
Initial report		Week 7
Full draft of the dissertation		2 weeks before final deadline

Learning outcome	The markers will assess	Pages[1]	Hours spent
Learning outcome 1 Conduct a literature search using an appropriate range of information sources and produce a critical review of the findings.	* Range of materials; list of references * The literature review/exposition/background information chapter		(for example, 200 hours)
Learning outcome 2 Demonstrate professional competence by sound project management and (a) by applying appropriate theoretical and practical computing concepts and techniques to a non-trivial problem, or (b) by undertaking an approved project of equivalent standard.	* Evidence of project management (Gantt chart, diary, etc.) * Depending on the topic: chapters on design, implementation, methods, experiments, results, etc.		(for example, 200 hours)
Learning outcome 3 Show a capacity for self-appraisal by analysing the strengths and weakness of the project outcomes with reference to the initial objectives, and to the work of others.	* Chapter on evaluation (assessing your outcomes against the project aims and objectives) * Discussion of your project's output compared to the work of others.		(for example, 100 hours)
Learning outcome 4 Provide evidence of the meeting learning outcomes 1-3 in the form of a dissertation which complies with the	* Is the dissertation well-written (academic writing style, grammatical), spell-checked, free of typos, neatly formatted.		

[1] Please note the page numbers where evidence of meeting the learning outcome can be found in your dissertation.

requirements of the School of Computing both in style and content.	* Does the dissertation contain all relevant chapters, appendices, title and contents pages, etc. * Style and content of the dissertation.	(for example, 80 hours)
Learning outcome 5 Defend the work orally at a viva voce examination.	* Performance * Confirm authorship	1 hour

Have you previously uploaded your dissertation to Turnitin? Yes/No

Has your supervisor seen a full draft of the dissertation before submission? Yes/No

Has your supervisor said that you are ready to submit the dissertation? Yes/No

Authorship Declaration

I, Diego M.R. Sanchez, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed:

DRS

Date: 14th of August 2020

Matriculation no: 40430644

General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below *one* of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

Diego

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

Natural Language Processing (NLP) comprises a set of methods for making human language accessible to computers. It is focused on the design and analysis of computational algorithms and representations for processing human language.

The General Language Understanding Evaluation benchmark (GLUE) is a collection of datasets used for training, evaluating and analysing NLP models relative to one another, with the goal of driving "research in the development of general and robust natural language understanding systems." (Wang, 2019).

This project explores different approaches to NLP and tests the performance of models with and without attention mechanisms in a set of tasks proposed by GLUE. The model implemented is a Siamese LSTM that yields very competitive results and even outperforms more complex attention based architectures in one of the settings.

The results show that, although attention mechanisms improve overall performance, the improvement is not as marked as could have been expected.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor, Dr. Kevin Sim, for the patience, wisdom and sense of humour he has shared all these months. I will also like to thank Dr. Simon Wells and Richard Plant for the great conversations and ideas that helped me in starting this journey.

To the friends that make me smile when things turn blue and life looks miserable. Special thanks to Luka, who has shared a considerable part of this process, and Steven, who has been pushing my intellectual boundaries for a number of years now.

I'd like to remember my family for believing in me, and Katerina, for showing me the way. I couldn't make it without all of you, thank you from my heart.

Edinburgh, 14th of August 2020.

1 Introduction

Natural Language Processing (NLP) comprises a set of methods for making human language accessible to computers. It is focused on the design and analysis of computational algorithms and representations for processing human language.

The goal of NLP is to succeed in activities that require human language abilities like extracting information from texts, translating between languages, answering questions, holding a conversation, etc. (Eisenstein, 2018).

NLP aims at understanding human language and processing it in a way that is not exclusive to a single task. The General Language Understanding Evaluation benchmark (GLUE) is a collection of datasets used for training, evaluating and analysing NLP models relative to one another, with the goal of driving "research in the development of general and robust natural language understanding systems." (Wang, 2019).

State-of-the-art approaches to NLP tend to ensemble networks like Recurrent Neural Networks, Long Short-Term Memory (LSTM), or Gated Recurrent (GRU) neural networks in particular. The best architectures make use of encoder and decoder structures that are connected through **attention mechanisms** (Vaswani, 2017). Attention mechanisms enable contextualized representations of words by calculating the probability of two given words being linked.

The aim of this project is to design a model architecture capable of dealing with a set of tasks proposed by the GLUE benchmark. The main questions the project will try to answer are:

- How much do attention mechanisms contribute to the performance of a model in common NLP tasks.

- What is the trade-off between computing effort and performance in common NLP technologies? Can the processing time be reduced without significantly losing performance?

In achieving this, state-of-the-art technologies will be utilized, with a specific focus on lightweight implementations that enable low computational power.

2 Literature Review.

This section starts by disambiguating the field where this project sits on, namely, Computational Linguistics and Natural Language Processing. It continues by reviewing common Machine Learning techniques applied in NLP and their progression across time from early developments to contemporary technologies.

Finally, the benchmark for testing those technologies is introduced, and other approaches to benchmarking discussed.

2.1 Computational Linguistics and Natural Language Processing

Perhaps the first attempt to make computers understand language can be attributed to Alan Turing, who made a simple question back in 1950, 'Can machines think?'. Sharply enough, he continued by attempting to define what is meant by 'machine' and 'think'. Without going into the nuances of Turing's thought, from his proposal 'The Imitation Game' (Turing, A. 1950), it seems clear that what is understood by thinking is the ability for a computer to process human language. Even more, to trick a human observer into thinking that a computer producing language is indeed a human being. This game is known as the Turing Test.

Since the Turing Test inception, computational linguistics has developed building up upon theories from theoretical linguistics, philosophical logic, psycholinguistics, and computer science.

Computational Linguistics (CL) and Natural Language Processing (NLP) have often been used interchangeably. However, it can be said that CL is a broader discipline and that NLP is derived from a practical application of some of its principles. More concretely, as Charniak proclaimed in the 50 anniversary of the Dartmouth Artificial Intelligence Conference in 2006: "Natural Language Processing is Now Statistical Natural Language Processing. Logicist AI is moribund, and the statistical approach is the only promising game in town – for the next 50 years." (Bringsjord & Govindarajulu, 2020).

2.1.1 Computational Linguistics

> "Human knowledge is expressed in language. *So computational linguistics is very important.*" –Mark Steedman, ACL Presidential Address (2007).

Computational linguistics is the scientific discipline concerned with how written and spoken language can be understood from a computational perspective (Schubert, 2014). Its goals are broad and varied, ranging from machine translation, question answering, text summarization, and others.

Language serves to convey meaning and meaning can be represented in different ways. Following Schubert, it can be said that computational linguistics have taken three different approaches to meaning representation: the logicist approach, the cognitive science approach, and the statistical approach.

This project focus on the statistical approach for being the one receiving the most interest in recent times. However, it is important to note that the field was predominantly dominated by the logicist oriented perspective until early 2010. The reason for the statistical approach not being so developed by then was the lack of a "large gold standard annotated corpora, integrating various deep semantic phenomena rather than focusing on one" (Bos, 2011). A more detailed explanation of computational semantic approaches can be found in Bos 2011.

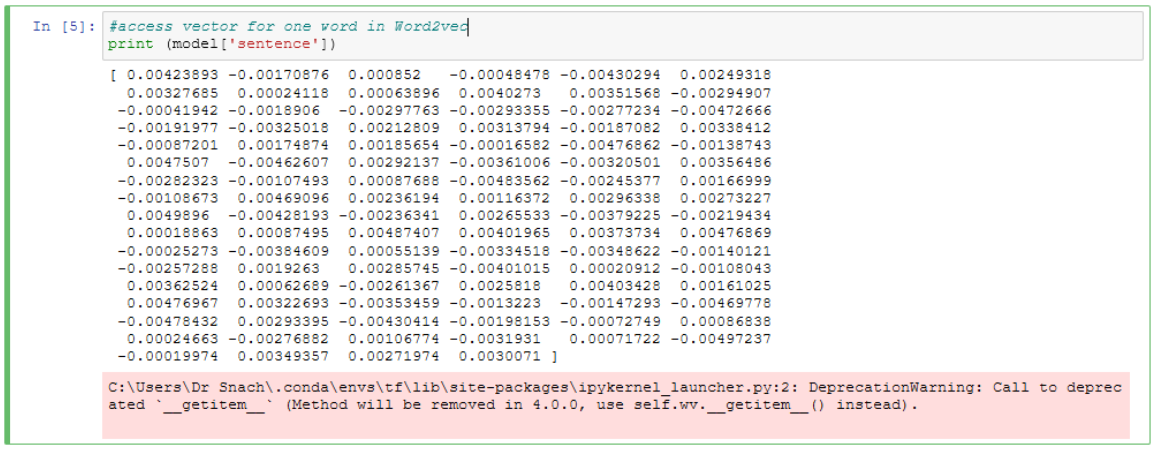
2.1.2 Natural Language Processing

This project is framed in the field of **Natural Language Processing** (NLP), which comprises a set of methods for making human language accessible to computers. It is focused on the design and analysis of computational algorithms and representations for processing human language. Similar to Computational Linguistics, the goal of NLP is to succeed in activities that require human language abilities like extracting information from texts, translating between languages, answering questions, holding a conversation, etc. (Eisenstein, 2018). Achieving this goal requires computers to 'understand' human language.

The representations for processing human language stem from statistical semantics, which is based on the **distributional hypothesis** formulated by Zellig Harris (Harris, 1954). It is often stated as "words which are similar in meaning occur in similar contexts" (Rubenstein & Goodenough, 1965). The general idea is that there is a correlation between distributional similarity and meaning similarity of words (Sahlgren, 2006).

Building on this hypothesis emerge distributed representations of words in a vector space. These representations are known as **word embeddings** and are obtained by calculating how probable is for a word to follow another set of words, which is its context. And the other way around, given a certain word, which context is more likely for it. Common word embedding algorithms include Word2vec (Mikolov et al., 2013) and GloVe (Pennington, Socher, & Manning, 2014).

The following is the embedding representation of the word 'sentence' in a 100-dimensional vector space using Word2vec.



```
In [5]: #access vector for one word in Word2vec
        print (model['sentence'])

[ 0.00423893 -0.00170876  0.000852   -0.00048478 -0.00430294  0.00249318
  0.00327685  0.00024118  0.00063896  0.0040273   0.00351568 -0.00294907
 -0.00041942 -0.0018906  -0.00297763 -0.00293355 -0.00277234 -0.00472666
 -0.00191977 -0.00325018  0.00212809  0.00313794 -0.00187082  0.00338412
 -0.00087201  0.00174874  0.00185654 -0.00016582 -0.00476862 -0.00138743
  0.0047507  -0.00462607  0.00292137 -0.00361006 -0.00320501  0.00356486
 -0.00282323 -0.00107493  0.00087688 -0.00483562 -0.00245377  0.00166999
 -0.00108673  0.00469096  0.00236194  0.00116372  0.00296338  0.00273227
  0.0049896  -0.00428193 -0.00236341  0.00265533 -0.00379225 -0.00219434
  0.00018863  0.00087495  0.00487407  0.00401965  0.00373734  0.00476869
 -0.00025273 -0.00384609  0.00055139 -0.00334518 -0.00348622 -0.00140121
 -0.00257288  0.0019263   0.00285745 -0.00401015  0.00020912 -0.00108043
  0.00362524  0.00062689 -0.00261367  0.0025818   0.00403428  0.00161025
  0.00476967  0.00322693 -0.00353459 -0.0013223  -0.00147293 -0.00469778
 -0.00478432  0.00293395 -0.00430414 -0.00198153 -0.00072749  0.00086838
  0.00024663 -0.00276882  0.00106774 -0.0031931   0.00071722 -0.00497237
 -0.00019974  0.00349357  0.00271974  0.0030071 ]

C:\Users\Dr Snach\.conda\envs\tf\lib\site-packages\ipykernel_launcher.py:2: DeprecationWarning: Call to deprec
ated `__getitem__` (Method will be removed in 4.0.0, use self.wv.__getitem__() instead).
```

Figure 1. Word embedding for 'sentence' in Word2vec with 100 dimensions

These embeddings are able to capture a substantial amount of information about the words, and the relations about them. The patterns of this relations can be represented linearly as shown in Figure 2.

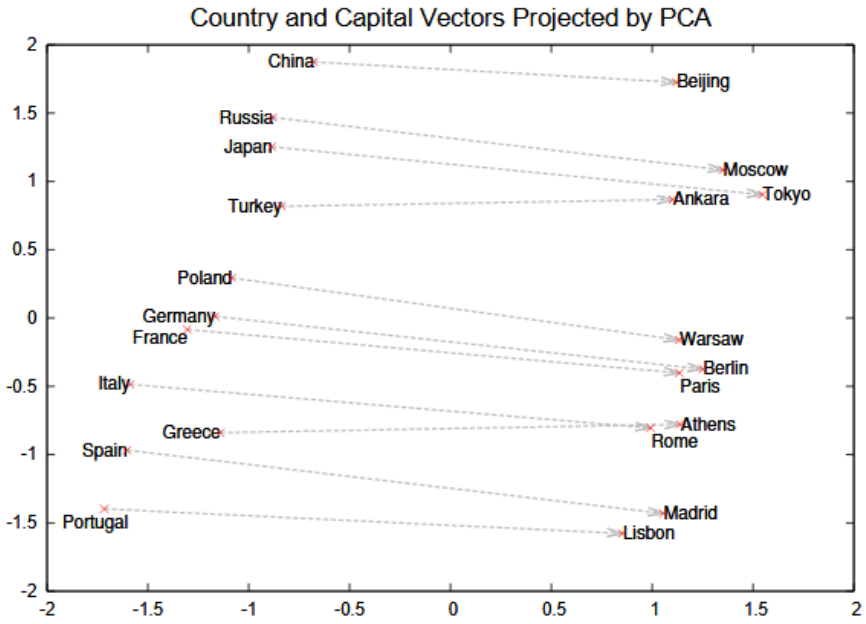


Figure 2. Embeddings for countries and capital cities from Mikolov et al., 2013

These examples are based on two-dimensional projections of the word embedding for countries and capitals. No labeled data about the nature of their relationships was required to identify the underlying structure. Distributional statistics have a striking ability to capture lexical semantic relationships such as analogies (Eisenstein, 2018).

A step further in this direction is represented by the Universal Sentence Encoder (Cer et al., 2018), a model for encoding sentences into embedding vectors that outperforms word level embeddings in many NLP tasks.

2.2 Machine Learning techniques applied to NLP

Progress in NLP has been possible through advances in many disciplines, but particularly in the subfield of Machine Learning (ML). ML can be divided into three major areas: supervised learning, unsupervised learning, and reinforcement learning. This project is mostly concerned with supervised learning, although some methods from unsupervised learning will be utilised, like word embeddings.

Machine Learning can be defined as the scientific discipline concerned with building systems that improve their performance on a given task after seeing examples of ideal performance on that task.

In regards to what a machine actually 'learns', Mitchell provides the definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."(Mitchell, 1997).

Different methods have been proposed across history to deal with problems in the field of NLP. In the next sections, approaches for solving tasks in ML will be presented, focusing when appropriate in their applications to NLP. The review will start with simple models of the early developments and progress till state-of-the-art technologies that can surpass human performance for some NLP tasks.

2.2.1 Early developments

Neural networks are the dominant approach for non-linear classification in NLP today (Eisenstein, 2018). The simplest possible neural network is one that comprises one single neuron. A **neuron** in ML is a placeholder for a mathematical function.

The concept of neuron is rooted in the works of Santiago Ramón y Cajal and his research in neuroscience. The neuron was thus the most basic element in the brain, and its activity the result of communication among neurons. Working from these beginnings, Warren McCulloch and Walter Pitts introduced the concept of neuron to apply it in the logic field with the principle of

"all or none" (McCulloch & Pitts, 1943), which describes whether a neuron is activated or not. Their neuron looked like this:

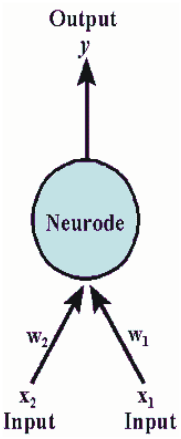


Figure 3. Neuron from McCulloch and Pitts in 1943

A more refined version was further described by Frank Rosenblatt, which incorporated the previous findings to develop the perceptron (Rosenblatt, 1962), an instrumental element in the later formation of neural networks. A perceptron can be represented as follows:

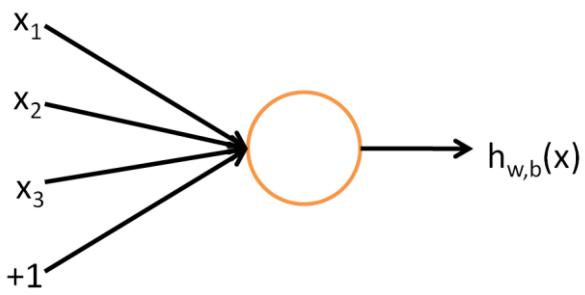


Figure 4. Single perceptron from Stanford.edu (Ng, 2019)

Where **x** are the input values fed into the neuron which process them through an activation function with parameters **w** and **b.** (w for weights, b for bias).

The **activation function** initially used by McCulloch & Pitts was the threshold function, following the principle of all or none. There are other activation functions that can be appreciated in the following graph:

Figure 5. Activation functions. Image from California State University Long Beach

Activation functions determine the output behaviour of each neuron in an artificial neural network. Although there are many possible choices, the Rectifier activation function (ReLU) (Glorot, Bordes, & Bengio, 2011), often outperforms other activation functions in deep neural networks and is the recommended option by Goodfellow et al. (2016). For negative inputs results in 0 and is linear for positive inputs. It looks like this:

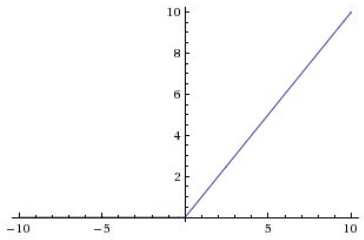


Figure 6. Rectified Linear Function. (Dansbecker, 2018)

Despite its simplicity, it performs well for many tasks, especially when combined with the dropout regularization technique. The technical advantages over other functions are that it doesn't involve expensive computations, and more importantly, it does not saturate (Goldberg, 2015).

2.2.2 Neural networks and Convolutional Neural Networks

Neurons and activation functions are the basic building blocks of any neural network. A **Neural Network** is put together by connecting simple neurons whose goal is to approximate some function. Figure 7 shows an example.

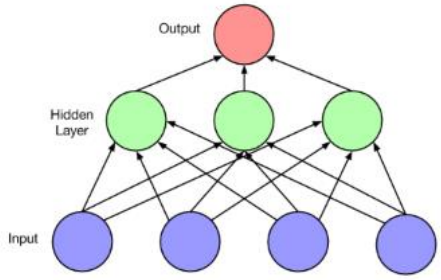


Figure 7. Neural Network (Lipton, Berkowitz, & Elkan, 2015)

The values of each neuron are calculated progressively from input to output and are a result of its activation functions and the input they receive. The network tries to approximate the parameters in the hidden layers that yield the best results, this approximation is its learning. The parameters reflect the importance of each neuron to the final output (Goldberg, 2015). Networks with more than one hidden layer are said to be *deep* networks, hence the name, *deep learning.*

This model is called feedforward because the information flows from input to output without feedback to the network itself. This is precisely its main limitation, as by doing so it is assuming independence among training and test examples. After each example is processed, the entire state of the network is lost. When examples are not independent but sequential, such as those of text or speech, this becomes a major problem (Lipton, Berkowitz and Elkan, 2015).

Convolutional Neural Networks (CNN) were originally used in computer vision, but they have shown good results when applied to NLP tasks. The mechanism behind a CNN is to apply a filter over the input, in this case sentences, and extract its most salient features. It is a way of condensing information while the filters are applied. The following image might provide some insight.

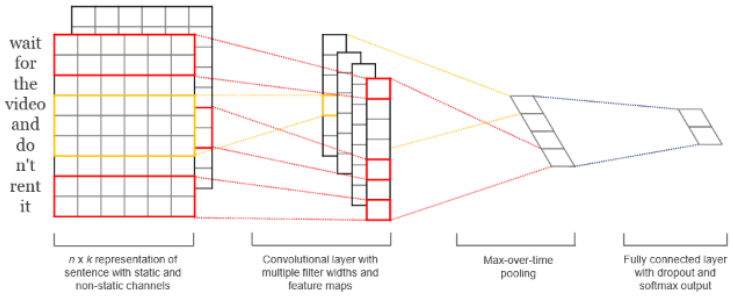


Figure 8. CNN architecture (Kim, 2014)

In the image the sentences are fed to the CNN and filters are applied with the intention of extract its most salient features, those that made them more likely to fall into one of the categories of the output.

Successful application of CNNs over NLP tasks include sentence modeling (Kalchbrenner et al., 2014), sentence classification (Kim, 2014) or other traditional NLP tasks (Collobert et al., 2011).

One of the problems that CNNs have is that, although they allow to encode large items in fixed-size vectors capturing their most salient features, they do so by sacrificing most of the structural information (Goldberg, 2016).

2.2.3 Recurrent Neural Networks and LSTMs

When the network is extended to include feedback connections is called **Recurrent Neural Network** (RNN) (Elman, 1990). The key point of RNN is that the recurrent connections allow a 'memory' of previous inputs to persist in the network's internal state, and thereby influence the network output (Graves and Pedrycz, 2009).

The networks are able to learn interesting internal representations that incorporate task demands. These representations reveal a rich structure, which allows them to be highly context-dependent and make them ideal for working with sequences like sentences or texts. (Elman, 1990). They have shown very strong results in different tasks like language modeling (Mikolov et al., 2013); machine translation (Sutskever, Vinyals, & Le, 2014) and sentiment analysis (Wang & Wang, 2015) among others.

RNN can be thought of as a series of networks linked together. From a graphical perspective, when the network is unrolled, it shows each output member as a function of the previous output. Each member of the output is produced using the same update rule applied to the previous outputs (Goodfellow et al. 2016). This phenomenon can be seen in Figure 9:

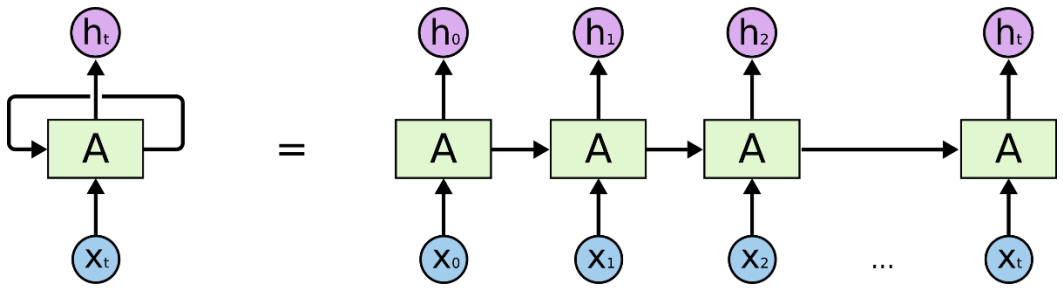


Figure 9. Unrolled RNN from (Olah, 2015)

The recurrent connections allow the network's hidden units to see its own previous output, so that the subsequent behaviour can be shaped by previous responses. These recurrent connections are what give the network memory (Elman, 1990).

At this point, one might wonder how much information can persist and for how long. Bengio showed the practical difficulties when the temporal contingencies present in the input and output sequences span long intervals, that is, when sequences are too long.

The basic problem is that gradients propagated over many stages tend to either vanish or explode. This effect is often referred to in the literature as the *vanishing gradient problem* (Hochreiter, 1991; Hochreiter et al., 2001a; Bengio et al., 1994). The capacity for holding information for RNN is thus relatively limited.

A crucial innovation to RNN is the **Long Short-Term Memory** (LSTM) (Hochreiter and Schmidhuber, 1997), which are capable of learning long-term dependencies and hold more information than standard RNN. They can also be referred to as Gated RNN, because they have mechanisms that act as gates, enabling some information to pass through the gate and update the network or discard it. The following image can provide some insight into its mechanism.

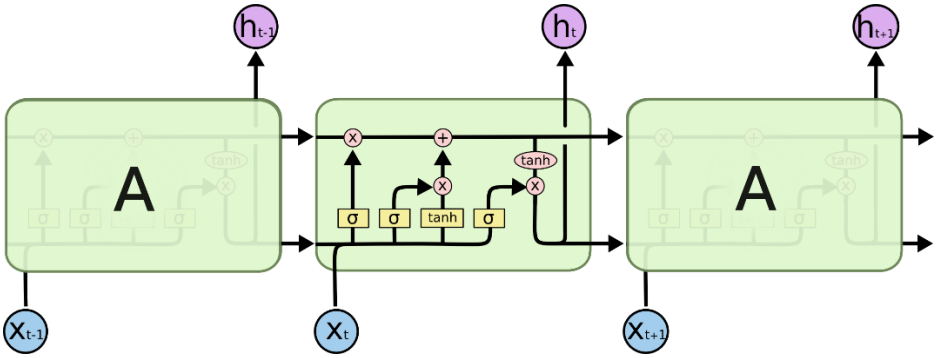


Figure 10. LSTM from (Olah, 2015)

LSTMs were developed precisely to cope with the vanishing gradient problem of RNN. When attempting to solve the proposed tasks in the GLUE benchmark, an adaptation of the LSTM will be utilized, a Siamese LSTM, discussed in section 5.4.

2.2.4 Encoder-Decoder

A successful model for processing sequences that makes use of LSTMs is the **Encoder-Decoder** or Sequence to Sequence (Sutskever, Vinyals, & Le, 2014) which was developed in the context of translation tasks. The LSTM's ability to learn data with long temporal dependencies makes it a natural choice for this application due to the considerable time lag between the inputs and outputs.

This method uses a multi-layered LSTM to map the input sequence to a vector of fixed dimensionality, and then another LSTM to decode the target sequence from the vector. One of its benefits is it can be trained to map input sequences to outputs where their length is unknown and variable. The idea behind this architecture is as follows:

- An encoder or input LSTM process the input sequence and emits a context C (or encoder vector)
- A decoder or output LSTM, conditioned by C, generates an output sequence.

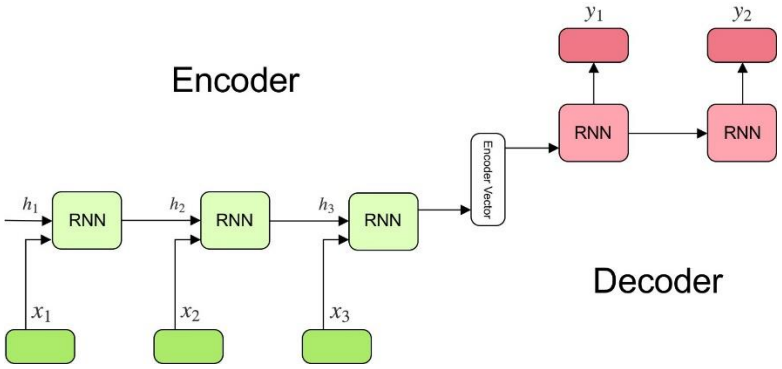


Figure 11. Encoder-Decoder mechanism. (Kostadinov, 2019)

Interestingly, the authors found that feeding the input in reverse order on the first LSTM improves results significantly. They argue that doing so introduces many short-term dependencies that make the optimization problem much easier (Sutskever, Vinyals, & Le, 2014).

A potential issue with this encoder-decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector (the context vector C) (Bahdanau, Cho, & Bengio, 2015).

2.2.5 Attention mechanism

Building on the encoder-decoder, Bahdanau conjectures that the use of this fixed-length vector is a bottleneck in the model architecture. To tackle this issue, they propose to extend it and to allow the model to search for parts of the input that are relevant for predicting the output word(s).

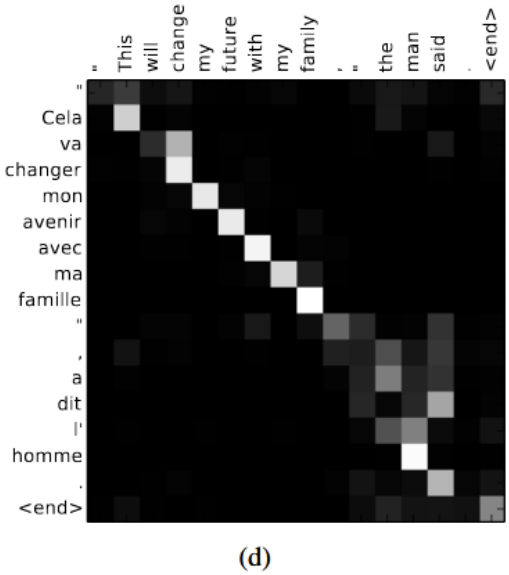
The most important feature of this approach is it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively while decoding the translation. By doing this, their model copes better with long sentences (Bahdanau, Cho, & Bengio, 2015).

The new architecture consists of a bidirectional RNN as an encoder and a decoder that searches through the input sentence while decoding in the translation. It does so with an alignment model that scores how well two given words match.

Intuitively, this implements a mechanism of **attention** in the decoder, as it focuses on the parts of the sentence that are relevant for its translation. With the decoder paying attention at the encoder, the encoder is relieved from having to encode all information in the source sentence into a fixed-length vector.

With this new approach, the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly. This has proven to increase the ability of the model to yield good results when translating long sentences.

Figure 12 shows graphically how the model aligns words in the translation, that is, it shows which words from the input sentence are relevant to the output translation. Each word(s) contributes to other word(s) in the translation.



(d)

Figure 12.Attention mechanism in action. (Bahdanau et al., 2015).

This architecture has pushed state-of-the-art performance for a number of NLP tasks (Bahdanau, Cho, & Bengio, 2015), and contributes to a better understanding of natural language in general. One of the limitations this model has shown is in handle unknown or rare words.

2.2.6 Transformer

Sequence transduction models are typically the ones used to solved translation tasks, where the order of words plays a crucial role. They are often built up with RNN and CNN and make use of an encoder and decoder mechanism, that creates a context vector(s) as previous sections have described.

Vaswani and colleagues proposed a new simple mechanism, the **Transformer,** based solely on attention mechanisms that gets rid of recurrence and convolutions, thus simplifying the architecture of previous models. (Vaswani et al., 2017).

Attention mechanisms, as previously described, are a powerful tool to calculate how two (or more) given words are link together, or dependent from each other. Self-attention is an attention mechanism directed to a single sequence from the input or output that enable to learn local dependencies in a single sequence. The following example can provide some insight.

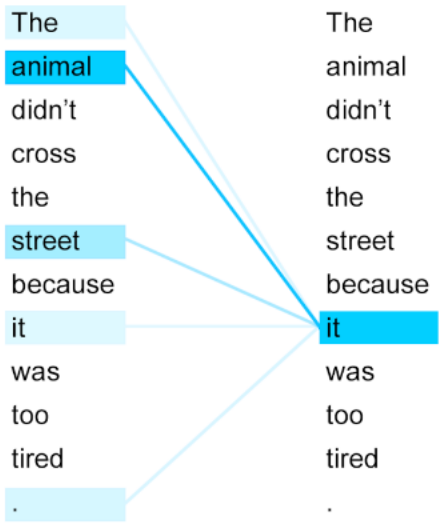


Figure 13. Encoder self-attention distribution for the word 'it' (Vaswani et al., 2017)

Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations.

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The Transformer uses multi-head attention in three different ways:

- As regular encoder-decoder from input to output
- With self-attention layers for the encoder
- With self-attention in the decoder

They follow the encoder-decoder architecture, generating a context vector that helps the decoder in emitting its output. The novelty they bring is to completely get rid of recurrence and convolutions. To make sense of the sequential data, they come up with a positional encoder that helps in preserving the order of the tokens in the sentences.

Vaswani's architecture is able to reduce the complexity of the models used in transduction tasks. They manage to parallelize a substantial amount of operations needed in their model, without losing performance, and they manage to push the state of the art in translation tasks (Vaswani et al., 2017).

2.3 General Language Understanding Evaluation benchmark

NLP aims at understanding human language and processing it in a way that is not exclusive to a single task. The General Language Understanding Evaluation benchmark (GLUE) is a collection of datasets used for training, evaluating and analysing NLP models relative to one another, with the goal of driving 'research in the development of general and robust natural language understanding systems.' It has been proposed in February 2019 by a few researchers leaded by Alex Wang and with the collaboration of Google's DeepMind (Wang et al., 2019).

While humans' ability to understand language is general and flexible, most NLP models struggle when coping with tasks for which they have not been explicitly trained. The GLUE benchmark aims at developing models that enable transfer learning from one task to another, making NLP systems that are more generalizable.

The collection consists of nine "difficult and diverse" task datasets designed to test the language understanding of a model. These tasks involve question answering, sentiment analysis, and textual entailment. This way, the benchmark favours models that can learn to represent linguistic knowledge transferrable across tasks.

The final task is to test the model against a small, manually curated test set for general analysis of the system performance, the **GLUE diagnostic dataset.** This dataset aims at testing the following linguistic phenomena: lexical semantics, predicate-argument structure, logic, and knowledge. It is provided as a tool to analyse the phenomena that a model might or might not capture, and thus expose a model's weaknesses, how it compares to other models and how it deals with specific aspects of linguistics.

In this project, a set of the proposed tasks will be tackled with different models that will hopefully shed light on the following two questions:

- How much attention mechanisms contribute to the performance on a single task.
- What is the trade-off between processing time and accuracy with comparable models? For example, a general model like BERT (Bidirectional Encoder Representations from Transformers), and a distilled version of it.

2.3.1 Tasks in the benchmark

The GLUE benchmark proposes nine English sentence understanding tasks, that cover different domains, data quantities and level of difficulty. The goal is to design a model capable of dealing with those tasks, which are divided as follows:

- Single sentence tasks: CoLA and SSST-2.
- Similarity and paraphrase tasks: MRPC, QQP and STS-B.
- Inference tasks: MNLI, QNLI, RTE, WNLI.

A brief description of the tasks can be seen in the following table:

Task	Description
Corpus of Linguistic Acceptability (CoLA)	The examples are phrases and the goal is to evaluate whether they are grammatical or not.
Stanford Sentiment Treebank (SST-2)	Consist of sentences taken from movie reviews and the task is to predict whether the sentiment is positive or negative
Microsoft Research Paraphrase Corpus (MRPC)	Provides sentence pairs and the goal is to decide if they are semantically equivalent
Quora Question Pairs (QQP)	A collection of question pairs where the task is to determine if they are semantically equivalent
Semantic Textual Similarity Benchmark (STS-B)	Consists of sentence pairs annotated with a similarity score from 1 to 5, where the goal is to predict the score.
Multi-Genre Natural Language Inference Corpus (MNLI)	Sentence pairs with textual entailment annotations. The task is to predict whether the first one entails the second, contradicts it or neither.
Stanford Question Answering Dataset (QNLI)	A refined version of the original SQuAD. Provides question-answering pairs where the answer of the question might be contained in the context paragraph or not.
Recognizing Textual Entailment (RTE)	A textual entailment task where the goal is to predict whether the second sentence entails the first or not.

Winograd Schema Challenge (WNLI)	A refined version of the original Winograd Schema where two sentences are provided with ambiguous pronouns. The task is to predict if the sentence with an ambiguous pronoun is entailed by the original sentence.

Table 1. GLUE tasks

Once a system has dealt with the proposed tasks, the results can be uploaded to the website gluebenchmark.com for scoring and join a leaderboard based on its relative performance. The tasks are evaluated on accuracy and, when labels are imbalanced, on F1 score. Pearson and Matthew's correlation are also measured STS-B and CoLA respectively. They will be detailed in section 5.2.

2.3.2 Baselines

The authors propose baseline models and facilitate the code as starting point. The architecture they propose is a two-layer BiLSTM (a bidirectional LSTM as seen in section 2.2.3) encoder, with max-pooling and the 300 dimensions Glove embedding (as discussed in section 2.1.2). The classifier they use is an MLP with 512 dimensions hidden layer. They also proposed a variant of the model that makes use of attention mechanisms, and several variants based on recent pretraining methods. The results they obtain with these models can be seen in Figure 14.

Model	Avg	Single Sentence		Similarity and Paraphrase			Natural Language Inference			
		CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	WNLI
Single-Task Training										
BiLSTM	63.9	15.7	85.9	69.3/79.4	81.7/61.4	66.0/62.8	70.3/70.8	75.7	52.8	**65.1**
+ELMo	66.4	**35.0**	90.2	69.0/80.8	85.7/65.6	64.0/60.2	72.9/73.4	71.7	50.1	**65.1**
+CoVe	64.0	14.5	88.5	73.4/81.4	83.3/59.4	67.2/64.1	64.5/64.8	75.4	53.5	**65.1**
+Attn	63.9	15.7	85.9	68.5/80.3	83.5/62.9	59.3/55.8	74.2/73.8	77.2	51.9	**65.1**
+Attn, ELMo	66.5	**35.0**	90.2	68.8/80.2	**86.5/66.1**	55.5/52.5	**76.9/76.7**	76.7	50.4	**65.1**
+Attn, CoVe	63.2	14.5	88.5	68.6/79.7	84.1/60.1	57.2/53.6	71.6/71.5	74.5	52.7	**65.1**
Multi-Task Training										
BiLSTM	64.2	11.6	82.8	74.3/81.8	84.2/62.5	70.3/67.8	65.4/66.1	74.6	57.4	**65.1**
+ELMo	67.7	32.1	89.3	**78.0/84.7**	82.6/61.1	67.2/67.9	70.3/67.8	75.5	57.4	**65.1**
+CoVe	62.9	18.5	81.9	71.5/78.7	84.9/60.6	64.4/62.7	65.4/65.7	70.8	52.7	**65.1**
+Attn	65.6	18.6	83.0	76.2/83.9	82.4/60.1	72.8/70.5	67.6/68.3	74.3	58.4	**65.1**
+Attn, ELMo	**70.0**	33.6	**90.4**	**78.0**/84.4	84.3/63.1	74.2/72.3	74.1/74.5	**79.8**	58.9	**65.1**
+Attn, CoVe	63.1	8.3	80.7	71.8/80.0	83.4/60.5	69.8/68.4	68.1/68.6	72.9	56.0	**65.1**
Pre-Trained Sentence Representation Models										
CBoW	58.9	0.0	80.0	73.4/81.5	79.1/51.4	61.2/58.7	56.0/56.4	72.1	54.1	**65.1**
Skip-Thought	61.3	0.0	81.8	71.7/80.8	82.2/56.4	71.8/69.7	62.9/62.8	72.9	53.1	**65.1**
InferSent	63.9	4.5	85.1	74.1/81.2	81.7/59.1	75.9/75.3	66.1/65.7	72.7	58.0	**65.1**
DisSent	62.0	4.9	83.7	74.1/81.7	82.6/59.5	66.1/64.8	58.7/59.1	73.9	56.4	**65.1**
GenSen	66.2	7.7	83.1	76.6/83.0	82.9/59.8	**79.3/79.2**	71.4/71.3	78.6	**59.2**	**65.1**

Figure 14. GLUE tasks baseline performance (Wang et al., 2019)

In a similar manner, researchers from New York University conduct experiments in order to provide a measure of **human performance** in the benchmark (Nangia & Bowman, 2019). They do so by hiring non-expert annotators and confronting them with the tasks after a brief training exercise. They compare their performance with recent high achieving models, and asses how these models perform when data training is scarce, in the same way that human annotators are confronted with the tasks after just a brief training. The outcome of their experiments is shown in the following table.

	Avg	Single Sentence CoLA	SST-2	Sentence Similarity MRPC	STS-B	QQP	Natural Language Inference MNLI	QNLI	RTE	WNLI
Training Size	-	*8.5k*	*67k*	*3.7k*	*7k*	*364k*	*393k*	*108k*	*2.5k*	*634*
Human	**87.1**	**66.4**	**97.8**	80.8/86.3	**92.7/92.6**	80.4/59.5	**92.0/92.8**	91.2	**93.6**	**95.9**
BERT	80.5	60.5	94.9	85.4/89.3	87.6/86.5	**89.3/72.1**	86.7/85.9	**92.7**	70.1	65.1
BigBird	83.9	65.4	95.6	**88.2/91.1**	89.5/89.0	**89.6/72.7**	87.9/87.4	95.8*	85.1	65.1
Δ_{bert} (-)	6.6	5.9	2.9	-4.6/-3.0	5.1/6.1	-8.9/-12.6	5.3/6.9	-1.5	23.5	30.8
Δ_{bird} (-)	3.2	1.0	2.2	-7.4/-4.8	3.2/3.6	-9.2/-13.2	4.1/5.4	-4.6*	8.5	30.8
Performance on subset with 5-way annotator agreement										
Human	93.7	83.6	100.0	90.2/93.6	98.9/94.7	89.4/74.1	98.5/99.2	95.1	97.4	97.5
BERT	83.5	69.2	97.5	88.9/92.7	95.8/82.3	92.5/78.0	96.4/90.8	93.6	73.0	59.3
Δ (-)	10.2	14.4	2.5	1.3/0.9	3.1/12.4	-3.1/-3.9	2.1/8.4	1.5	24.4	38.2
BERT fine-tuned on less data										
BERT-5000	75.8	57.6	92.0	85.4/89.3	87.1/85.8	82.2/61.0	76.4/76.9	89.2	69.2	65.1
BERT-1000	70.7	49.0	90.4	78.5/84.3	83.6/82.3	77.8/55.8	66.5/68.3	86.6	65.6	65.1
BERT-500	68.5	37.2	88.1	74.0/80.7	77.3/75.2	75.4/51.2	61.8/63.0	85.7	61.5	65.1

Figure 15. Human performance Vs High achieving models (Nangia et al., 2019)

These results show that the best performing statistical models are not far behind human execution but, more interestingly, they show significant drop in performance when training is scarce, thus validating the intuition that these models require huge amount of data in order to achieve high results.

The following image shows how, as of this writing in July 2020, human performance has been surpassed already and achieves a modest 13th place in the GLUE leaderboard.

	Rank	Name	Model	URL	Score
+	1	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6
	2	ERNIE Team - Baidu	ERNIE		90.4
+	3	Alibaba DAMO NLP	StructBERT		90.3
	4	T5 Team - Google	T5		90.3
	5	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9
+	6	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		89.7
+	7	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4
+	8	Huawei Noah's Ark Lab	NEZHA-Large		89.1
+	9	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4
	10	Junjie Yang	HIRE-RoBERTa		88.3
	11	Facebook AI	RoBERTa		88.1
+	12	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6
	13	GLUE Human Baselines	GLUE Human Baselines		87.1
	14	Stanford Hazy Research	Snorkel MeTaL		83.2

Figure 16. GLUE leaderboard in July 2020

2.3.3 Other evaluation approaches

This project is focused on NLP benchmarking, however, a broader perspective is taken when measuring intelligence with the aim of designing more intelligent systems. As NLP sits in the field of Computational Linguistics (CL), CL sits in the broader field of Artificial Intelligence (AI).

Satisfactorily benchmarking intelligence in AI is still a pending task as Francis Chollet (author of the popular library Keras) documents (Chollet, 2019). Most attempts in that direction tend to measure skill in a narrow task, often videogames or board games such as chess. Chollet argues that a wider definition is needed for guiding developments towards more generalizable systems, capable of performing tasks they haven't been explicitly trained for. The goal would be to measure ability, rather than skills in a particular task.

The approach Chollet proposes is to develop broad AI systems, with a degree of generality comparable to human intelligence, that is, able to handle tasks that differ from previously encountered ones. Thus, the intelligence of a system can be defined as: "… a measure of its skill-acquisition efficiency over a scope of tasks, with respect to priors, experience, and generalization difficulty" (Chollet, 2019). A highly intelligent system would be one that can generate valuable skilled solution programs for tasks that feature high uncertainty and that require high generalization.

With this definition of intelligence in mind, they introduce the Abstraction and Reasoning Corpus (ARC), a general artificial intelligence benchmark targeted at both humans and AI systems. The benchmark follows the psychometric tradition of intelligence tests and focuses on measuring a broad form of generalization. It consists of 1 thousand unique tasks that seek to measure generalization as previously described. The following is an example where the goal is implicitly set to complete the symmetrical pattern.

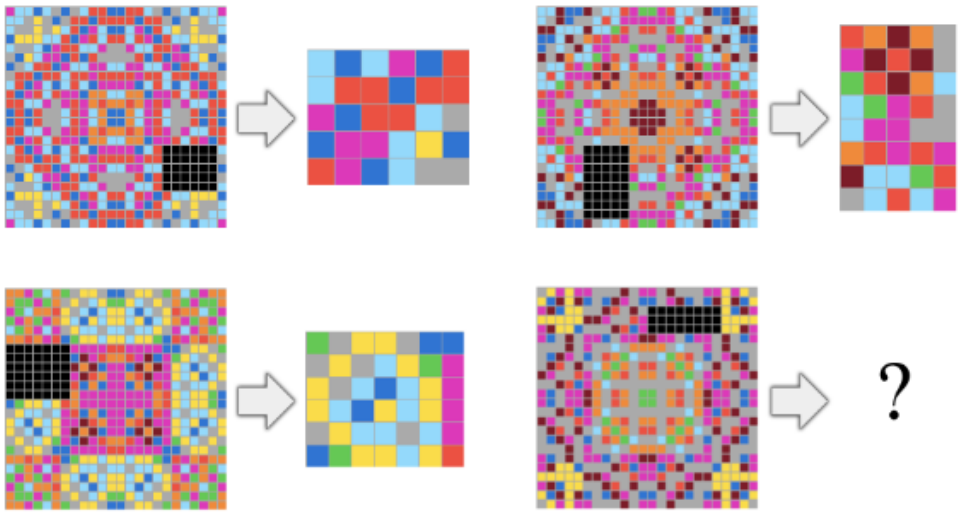


Figure 17. Example task from ARC dataset (Chollet, 2019)

The ARC benchmark is a great step towards creating more intelligent systems and ultimately get closer to an affirmative answer to Turin's question about whether machines can think. However, it is beyond the scope of this project to design a system able to deal with its proposed tasks. Instead, the focus will be on the GLUE benchmark, which also aims at designing more generalizable systems, just in the narrower perspective of NLP.

It is important to note that as of May 2020, a refined version of GLUE, SuperGLUE, has been made available for the general community. However, this project will still be using GLUE as it has been more tested and will allow for better comparisons.

2.4 SUMMARY

This section started defining what is understood by Natural Language Processing and has reviewed common architectures in ML and NLP. The benchmark that would be referenced in the experiments of this project was introduced.

The techniques' review started with the simplest perceptron and progressed till state-of-the-art technologies that make use of attention mechanisms and refined versions of RNN. Ultimately, the most complex architectures are simple neuron configurations stacked one after the another to produce more elaborated systems. Figure 18 provides some insight about the progression across time in the field.

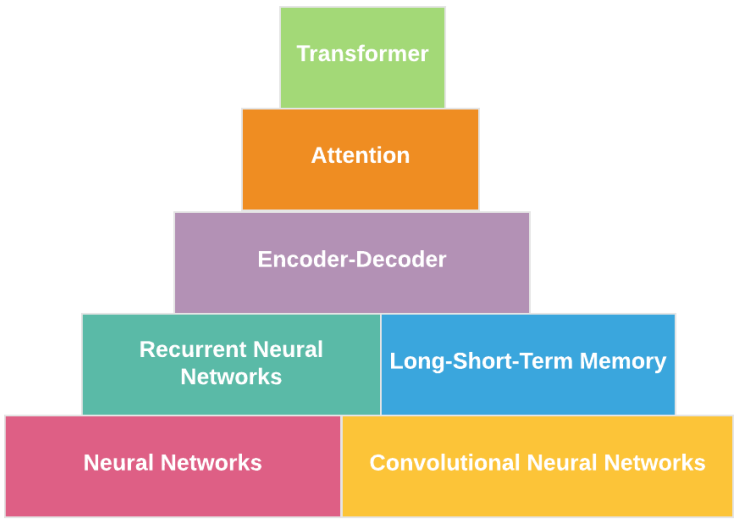


Figure 18. Machine Learning architectures

On the second half of the section, the GLUE benchmark was described, and baseline results were provided for both ML models and human performers. In the remaining part of the project, the focus would be in single task training for tasks related to Single Sentence and Similarity and Paraphrase.

Model	Avg	Single Sentence CoLA	SST-2	Similarity and Paraphrase MRPC	QQP	STS-B
				Single-Task Training		
BiLSTM	63.9	15.7	85.9	69.3/79.4	81.7/61.4	66.0/62.8
+ELMo	66.4	**35.0**	90.2	69.0/80.8	85.7/65.6	64.0/60.2
+CoVe	64.0	14.5	88.5	73.4/81.4	83.3/59.4	67.2/64.1
+Attn	63.9	15.7	85.9	68.5/80.3	83.5/62.9	59.3/55.8
+Attn, ELMo	66.5	**35.0**	90.2	68.8/80.2	**86.5/66.1**	55.5/52.5
+Attn, CoVe	63.2	14.5	88.5	68.6/79.7	84.1/60.1	57.2/53.6

Figure 19. Single-Task Training baselines (Wang et al., 2019)

The highlighted results are obtained with the BiLSTM model with attention mechanism provided by the authors. Section 4 will compare these results with the human performance and the Siamese LSTM model that will be implemented. This would help in shedding light on the question of how much attention mechanisms contribute to the performance of a model. Human performance in the set of tasks proposed can be seen in Figure 20.

	Avg	Single Sentence CoLA	SST-2	Sentence Similarity MRPC	STS-B	QQP
Training Size	-	*8.5k*	*67k*	*3.7k*	*7k*	*364k*
Human	**87.1**	**66.4**	**97.8**	80.8/86.3	**92.7/92.6**	80.4/59.5

Figure 20. Single-Task Human performance. Adapted from Nangia et al., 2019

3 Methodology

This chapter reviews different frameworks commonly used in ML tasks. It defines the metrics used in the experiments of this project and describes the typical steps to follow when tackling an NLP task.

The last section presents the implementation that have been selected, followed by a simple validation exercise to prove its effectiveness.

3.1 The choice of framework

There exist a growing number of deep learning frameworks in both open software and the commercial markets, and there is not a clear winner in terms of accuracy or performance. As Wu documents (Wu et al., 2019), the selection of a framework can be an overwhelming task. From their paper "A comparative measurement of Deep Learning as a Service Framework" two facts can be inferred. First, Python is the most popular language for Deep Learning applications as of 2020. Second, the list of the most popular frameworks according to the authors is Tensorflow, Caffe, Torch, Theano, CNTK, Keras and Pytorch.

For this project, the choice was narrowed to two libraries, Pytorch and Keras, this one tightly integrated with Tensorflow. **Keras** is defined in its official documentation as an API designed for human beings, not machines. It is simple, flexible and powerful, and if minimizes the number of actions required for common use cases. Its integrated with Tensorflow, maintained by Google. In the words of his author, Francois Chollet:

> "The library was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research."

Tensorflow is an open source library particularly designed for working with deep neural networks and *machine intelligence* as their authors call it. The popularity among the ML community for these two libraries can be appreciated in the following image:

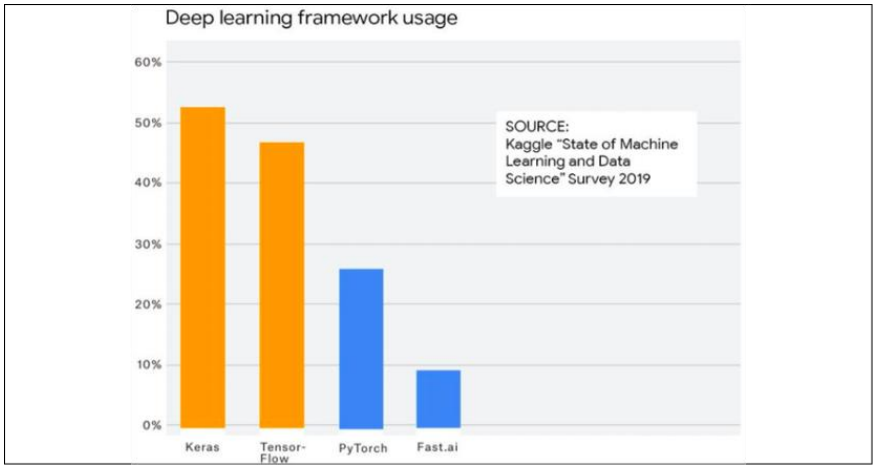


Figure 21. ML frameworks popularity (Kapoor, 2019)

Pytorch on the other side is based on Torch and developed by Facebook. It has a lower-level API that enables finer customization, at the cost of more complexity. For this project, Keras was the chosen framework for its great usability and easy customization. The simplicity and lack of verbosity of this approach can be observed in the following snippet of two similar implementations in both libraries:

Keras

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPool2D())
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

PyTorch

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3)
        self.conv2 = nn.Conv2d(32, 16, 3)
        self.fc1 = nn.Linear(16 * 6 * 6, 10)
        self.pool = nn.MaxPool2d(2, 2)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 6 * 6)
        x = F.log_softmax(self.fc1(x), dim=-1)
        return x
model = Net()
```

Figure 22. Keras Vs Pytorch. (Migdal & Jakubanis, 2018)

3.2 Metrics

The general metric for the tasks proposed in GLUE is accuracy, and when classes are imbalanced, the F-Score (F1). In STS-B and CoLA, the metrics are Pearson and Spearman correlation and Mathews correlation respectively. These metrics and their relevance are explained below.

Accuracy is defined as the proportion of examples for which the model produces the correct output. When labels are not evenly distributed, this measure can give an erroneous perception of the model. For instance, in a dataset where 98% of cases are positive and 2% of cases negative, a model would get a 98% accuracy by simply predicting all cases as positive.

In these scenarios, the **F-Score** provides a better estimation for the performance of a model. It calculates the harmonic mean of precision and recall and is used in such scenarios. Precision, recall and F1 are calculated as:

Precision = number of correct positive predictions / number of positive predictions
Recall = number of correct positive predictions / number of actual positives
F-score (or F1) = 2 x Precision x Recall / Precision + Recall

Table 2. F-Score measures

Pearson and Spearman **correlation** are measured in the STS-B task. In statistical terms, a correlation coefficient measures how well two variables correlate, that is, if one changes, the other also changes in the same or opposite direction. It can take values between -1 and 1, where 1 is perfect positive relation, -1 perfect negative relation and 0 no relation at all. The following image can give some insight:

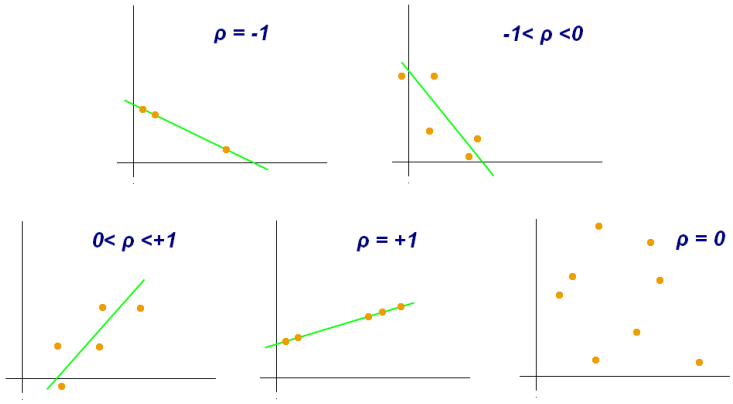


Figure 23. Linear correlation diagrams, (Kiatd, 2012)

Furthermore, the relation can be linear as in the previous diagrams or monotonic as is often the case. When the relation is linear, Pearson correlation is the preferred correlation measure, when is monotonic, Spearman is more accurate. The monotonic relation can be observed in the following image.

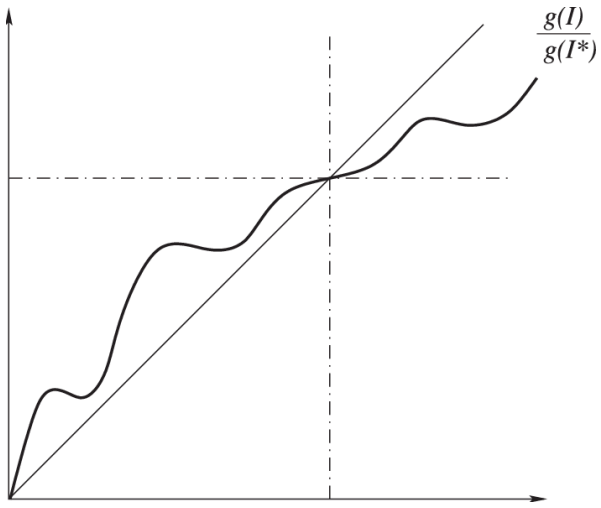


Figure 24. Monotonic function (Korobeinikov, 2005)

Finally, Matthews correlation is measured in the CoLA task, and is a metric that represents the quality of a binary classification when this is imbalanced. As with Pearson and Spearman, it can take the values from-1 to 1, being 0 no better prediction than random.

3.3 Tackling an NLP task

A fundamental aspect when dealing with an NLP task is how to represent language in a way that can be interpreted by a programming language, this is commonly referred as **feature representation.** When considering a neural network as a function **NN(x)** that takes an input **x** to produce and output vector **y,** the feature representation is concerned with how the input **x** is represented.

It's generally more desirable to provide a compact representation than a sparse one in common NLP tasks. Following (Goldberg, 2016), the structure of these tasks can be described as follows:

- Extract the core linguistic features, that is, ascribe a number to each of the words of the corpus in the dataset.
- For each word, retrieve the corresponding vector. This project uses Glove vectors by (Pennington, Socher, & Manning, 2014). The vectors provide word representations as described in section 2.1.
- Combine the vectors in different ways into the input x.
- Feed this input x into the Neural Network.

Feature representation can be easily done with the Keras Tokenizer in just a few steps. The following image can provide some insight about this process:

```
[ ]  # Converts sentences to sequences of integers
     # post pad them to max set to 40

     encoded_sent1 = tokenizer.texts_to_sequences(sentence1)
     padded_sent1 = pad_sequences(encoded_sent1, maxlen=max_len, padding='post')

     encoded_sent2 = tokenizer.texts_to_sequences(sentence2)
     padded_sent2 = pad_sequences(encoded_sent2, maxlen=max_len, padding='post')

     # Check
     print(sentence1[5])
     print(encoded_sent1[5])
     print(padded_sent1[5])

     At one of the three sampling sites at Huntington Beach  the bacteria reading came back at 160 on June 16 and at 120 on June 23 '
     [237, 94, 9, 3, 157, 5509, 1689, 16, 5510, 3386, 3, 5511, 2151, 558, 186, 16, 4231, 11, 401, 578, 10, 16, 2863, 11, 401, 641, 2]
     [ 237   94    9    3  157 5509 1689   16 5510 3386    3 5511 2151  558
       186   16 4231   11  401  578   10   16 2863   11  401  641    2    0
         0    0    0    0    0    0    0    0    0    0    0    0]
```

Figure 25. Feature representation in Keras

Then each word can be converted into a its vector representation by mapping the words present in the corpus with those from pretrained embeddings such as Glove. The process can be seen in the following Figure:

```
# Creates dictionary of word embeddings

glove_dir = './'

embeddings_index = {} #initialize dictionary
f = open(os.path.join('/content/gdrive/My Drive/glove.6B.100d.txt'))
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('Found %s word vectors.' % len(embeddings_index))
```

```
Found 400000 word vectors.
```

```
[ ] # Matches words/tokens from vocabulary to their embeddings

embedding_dimensions = 100

embedding_matrix = np.zeros((vocab_size, embedding_dimensions))

for word, i in tokenizer.word_index.items():
  embedding_vector = embeddings_index.get(word)
  if embedding_vector is not None:
    embedding_matrix[i] = embedding_vector
```

Figure 26. Embedding mapping

The embedding of a word was presented in section 2.1, its essentially a collection of numbers of **d** dimensions (typically 300). Once each word has been embedded, the embedding representation can be fed into a neural network for training. In the next section the neural network that will be used in this project will be described.

3.4 Implementation. Siamese LSTM

The chosen approach to deal with the tasks is the Siamese LSTM. It is particularly useful since many of the proposed tasks comprise double inputs in the form of two sentences, where the goal is to predict the relation among them. Where there is a single input, the chosen network is single LSTM for consistency.

The Siamese LSTM is based in the implementation that Mueller and Thyagarajan developed in 2016, which uses 2 identical LSTMs for each of the inputs and make use of Glove word embeddings with 300 dimensions. It exceeded **state-of-the-art in semantic similarity tasks** when published (Mueller & Thyagarajan, 2016). Schematically looks like this:

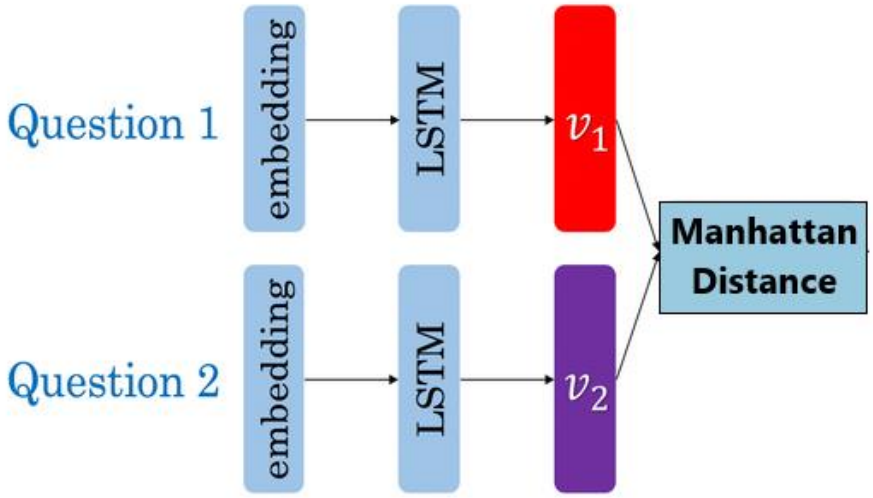


Figure 27. Siamese LSTM. Adapted from deeplearning.ai 2020

The intuition is that the Siamese LSTM encodes the embedded sentences into a fixed-size vector (similar to the context vector C of the encoder-decoder in section 2.2.4), that is of 50 dimensions in this case. It then computes the distance between the two representations which is used as a predictor of semantic similarity. The distance they compute is the Manhattan Distance, as opposed to the more common cosine similarity, as outperforms this and other measures in the authors' experiments (Mueller & Thyagarajan, 2016). Figure 28 shows the implementation in the experiments:

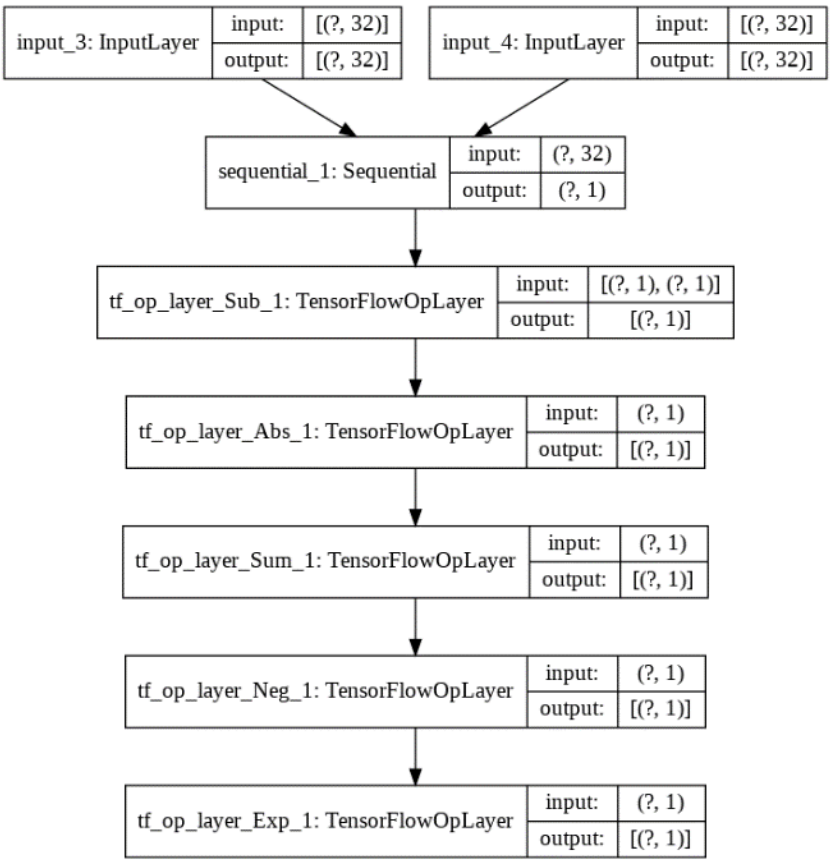


Figure 28. Siamaese LSTM implementation

Where each of the inputs (input 3 and 4 in the figure) comes from an LSTM, therefore the Siamese LSTM. Manhattan distance, also known as cityblock or taxicab, is that which computes the distance between two given points (in this case the input representation) along axes at right angles. The following image can provide some insight:

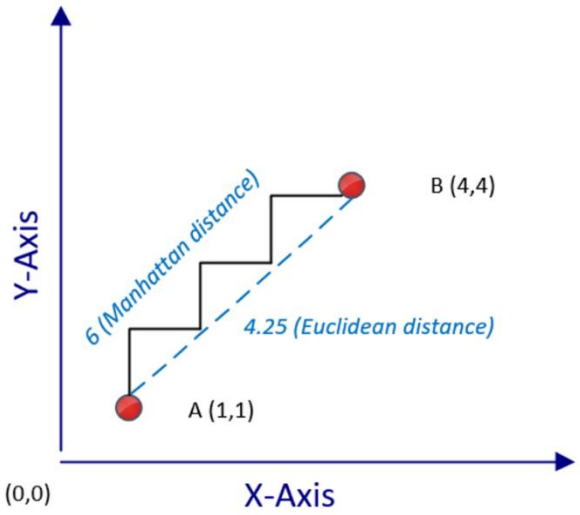


Figure 29. Manhattan Vs Euclidean Distance (Ahmad, 2020)

The network updates itself based on the difference between the predicted similarity and how it deviates from the ground truth.

3.4.1 Single LSTM

For tasks with a single input a single LSTM is used for consistency, with common function losses depending on the task. The model uses the Glove embedding with 300 dimensions and the LSTM has 50 hidden layers and 0.2 dropout. That is, at each pass, 20% of connections are discarded, which brings an overall better performance.

One of the benefits of these kind of models is they can be readily adapted to different ML tasks. To validate the model and show its efficiency, a common NLP task will be performed, the imbd reviews (Maas, 2011), which contains 25 thousand highly polar movie reviews. The single LSTM is defined as follows:

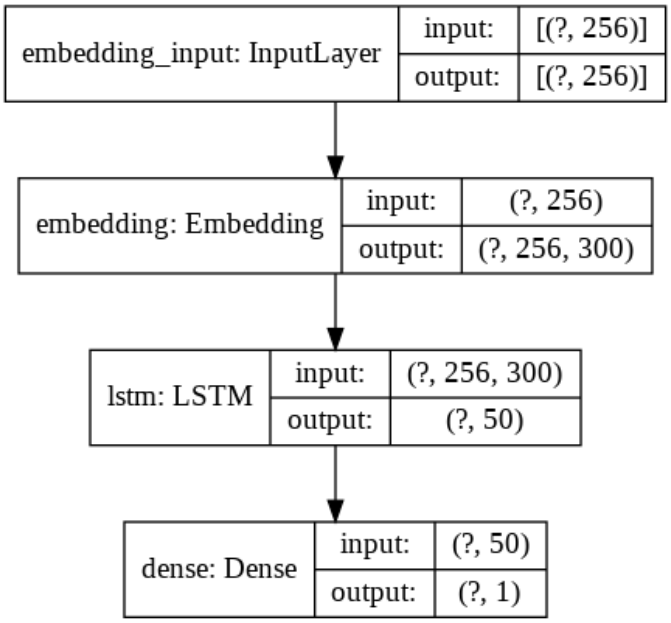


Figure 30. Single LSTM

The network takes as inputs 256 tokens for the max length of the sentences, process their embedding with 300 dimension (in the second layer), encodes them into 50 neurons of the LSTM and outputs into a single neuron that decides whether the review is positive or negative.

The following graph shows that after just two passes the network peaks at about 80% accuracy and keeps improving steadily. Training is stopped after 10 epochs achieving an accuracy in the validation set of ~ 86%.

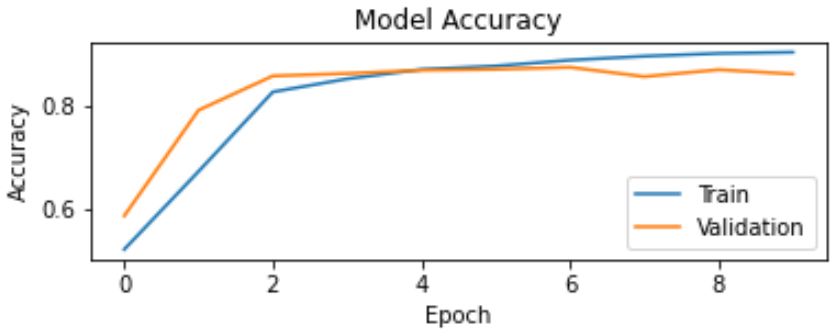


Figure 31. Single LSTM accuracy in imbd

Once the model is validated in a simple task, the results on the GLUE benchmark tasks will be presented.

4 Results

In the following chapter, the results of each task of the GLUE benchmark are described. A selection of 5 out of the 9 original tasks has been made, and the analysed ones are those of Single Sentence and Similarity and Paraphrase tasks. Inference tasks have been discarded to accommodate the time constraints and available resources for this project.

In all experiments, the embedding is Glove with 300 dimensions as described in section 2. Single LSTM is used for Single Sentence tasks and Siamese LSTM for Similarity and Paraphrase tasks. The architecture of the two models has been described in section 3.4.

Following common practice, LSTM layers have used tanh activation, and sigmoid activation for output layer. Dropout at 0.2 was used, that is, the bottom 20% neuron activations have been discarded. Loss functions were binary crossentropy for Single sentence tasks and mean squared error for Similarity tasks. Optimizer is Adam.

Each experiment has been conducted for 10 epochs, and the results are presented for a typical run. Some tasks would benefit from more training time, but this is kept to 10 epochs for consistency. There is a 12 hours limitation that Google Colab imposes on its users, and some tasks would render it insufficient if trained for more epochs, particularly QQP task.

Due to the stochastic nature of the tasks, experiments are conducted for 10 times and the range values with median and standard deviation are provided. The reported measures are as follows:

Task	Metric	Model
SST-2	Accuracy	Single LSTM
CoLA	Accuracy/Matthews Correlation	Single LSTM
MRPC	Accuracy	Siamese LSTM
QQP	Accuracy/F1	Siamese LSTM
STS-B	Pearson/Spearman Correlation	Siamese LSTM

Table 3. Metrics of the tasks

The results are presented for training, validation and test set on a typical experiment, accuracy plots are shown afterwards. Following usual convention, the plot will start at the yielded accuracy on first epoch, rather than in 0. This gives a clearer picture of its convergence.

Then a summary of 10 experiments is introduced. The summary was produced with the following custom function:

```
[ ]  # Run 10 experiments and plot results
```

```
[121] def make_model():
        tf.keras.backend.clear_session() # Starts session from scratch
        model = Sequential()
        model.add(embed)
        model.add(LSTM(n_hidden, dropout=0.2, activation='tanh', recurrent_dropout=0.2))
        model.add(Dense(1, activation='sigmoid'))
        # model.summary()
        model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                      loss=tf.keras.losses.binary_crossentropy,
                      metrics=['accuracy'])
        model.fit(padded_sentences,
                  label,
                  epochs=20,
                  batch_size=64,
                  class_weight=class_weights,
                  validation_split = 0.2,
                  verbose=2,
                  callbacks=[earlystop])
        results2 = model.evaluate(val_padded_sentences, val_label)
        prediction2 = model.predict(val_padded_sentences)
        true_pred = binary_prediction(prediction=prediction2)
        mat_coef = matthews_corrcoef(val2,true_pred)
        return mat_coef
```

```
[93] colection = [make_model() for _ in range (10)]
```

Figure 32. 10 runs validation function

The function repeats the experiments 10 times and yields the required metric for the task. In the rest of the section, each subsection would describe a task and give examples of its data. Then results are reported and conclusions are drawn.

4.1 Stanford Sentiment Treebank

The SST2 dataset (Socher et al., 2013) consists of sentences from movie reviews that are annotated as either positive or negative. There are 67.3 thousand reviews and the goal of the task is to predict the sentiment of the sentences. The following snippet shows some negative examples.

```
[6]  df_train.loc[df_train['label']==0]
```

	idx	label	sentence
0	16399	0	b'for the uninitiated plays better on video wi...
1	1680	0	b'like a giant commercial for universal studio...
4	27051	0	b', this cross-cultural soap opera is painfull...
6	54784	0	b'only masochistic moviegoers need apply . '
10	1456	0	b'revelatory nor truly edgy -- merely crassly ...
...	...	...	...
67342	32150	0	b"too much ai n't - she-cute baggage into her ...
67343	47134	0	b'despite some charm and heart , this quirky s...
67344	10487	0	b'even worse than its title '
67347	32842	0	b'the very definition of what critics have com...
67348	50006	0	b'barn-burningly bad movie '

29780 rows × 3 columns

Figure 33. Negative reviews from SST2

The Single LSTM is trained on the dataset and yields an accuracy on test set of ~82%. Results are shown in Table 5. In green appears accuracy for test set.

SST2				
	Training	**Validation**	**Test**	**Max Value**
Samples	53,879	13,469	872	-
Accuracy	92.51	91.60	81.99	92.51

Table 4. SST2 Results

The model performs well on training and validation but declines significantly on the test set. This is perhaps due to the scarcity of the test sample, just a small fraction of the validation test. Although is not reported, the loss in the test set is 0.47, while in training 0.19, which might be pointing that more training would be needed. An attempt to mitigate this circumstance is to reduce the batch size from 128 in training to 32 in test set, although it doesn't seem to improve accuracy.

Results seem to indicate that the model is reliable and that the task is accessible compared with others from the benchmark. The following plot shows how the model quickly arrives at 90% accuracy and then keeps improving in a steadier way. It might benefit for more training time, but to keep consistency with the rest of the tasks is left at 10 epochs.

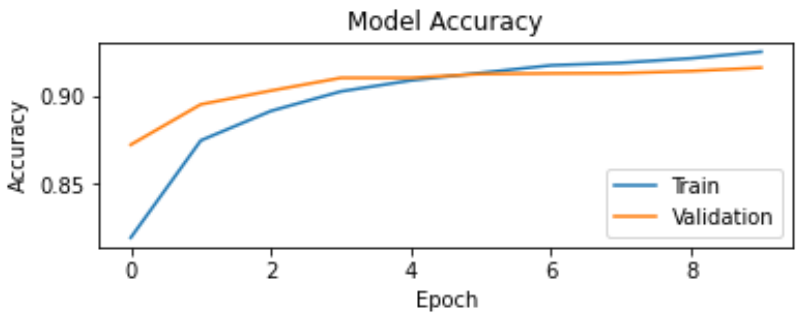


Figure 34. Plot results for SST2

For consistency in the results, experiments are run 10 times and average test results provided. These results are quite consistent and yield a median accuracy value of 82.51%, with a standard deviation of 0.00834, which suggests a very stable model. Figure 38 shows the results.

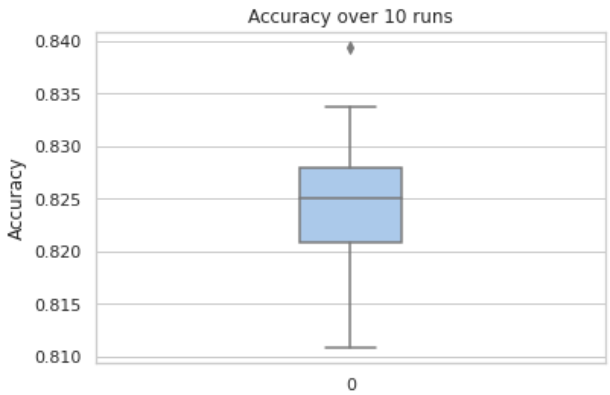


Figure 35. Accuracy over 10 runs for SST2

4.2 Corpus of Linguistic Acceptability

The CoLA dataset (Warstadt et al., 2018) consists of English sentences with annotations referring to whether they are grammatical or not. The labels are imbalanced, and the non-grammatical examples represent 30% of the total. The following image shows some examples of ungrammatical sentences.

```
df_train.loc[df_train['label']==0]
```

	idx	label	sentence
5	3017	0	b'The inspector analyzed the soundness in the ...
10	4352	0	b'There is more chemical substances involved i...
20	3365	0	b'In the corner lay a dog.'
23	6026	0	b'The cat was being eating.'
24	7931	0	b'The Greeks arrived all.'
...	...	...	...
8539	3682	0	b'John suddenly got the bus off.'
8544	7565	0	b'Mary believes that Bill saw herself.'
8546	4920	0	b'a pencil with that to write broke.'
8547	3619	0	b'It was in the park last night that the polic...
8549	8351	0	b'You said she liked yourself'

2528 rows × 3 columns

Figure 36. Ungrammatical examples from CoLA

It is important to note that the sentences would probably go unnoticed for the untrained eye. The model yields the following results in the dataset. Reported in green is the Matthews Correlation, the relevant metric for this task.

CoLA				
	Training	**Validation**	**Test**	**Max Value**
Samples	6840	1710	1043	-
Accuracy	71.15	69.49	68.93	71.15
Matthews Correlation			03.02	

Table 5. CoLA results

The model is not particularly accurate provided there's a class imbalance of 70/30. The results imply that only in a few cases the model is able to predict the negative label correctly. This gives an idea of the complexity of the task. The accuracy is plotted in the next image. It can be observed that the training accuracy is increasing, but not the validation, which suggests it might be overfitting.

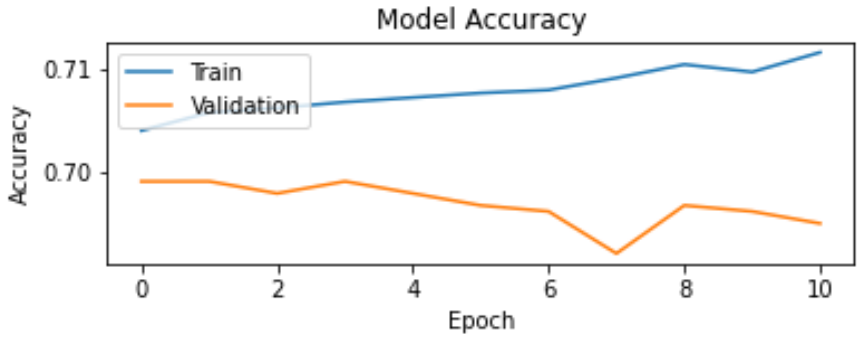


Figure 37. Accuracy for CoLA

Being the classes imbalanced, the Matthews correlation metric provides a more accurate picture of the performance of the model. Matthew's correlation is 3.02 positive from a maximum of 100. The model doesn't perform as expected but provides ground for further discussion.

Due to the stochastic nature of the experiments, these are conducted 10 times and average results and variance reported. The median obtained after 10 runs is similar to the first experiment, 1.94, with a standard deviation of 2.8134, which suggests the model is not very stable. Figure 35 shows the variability among experiments.

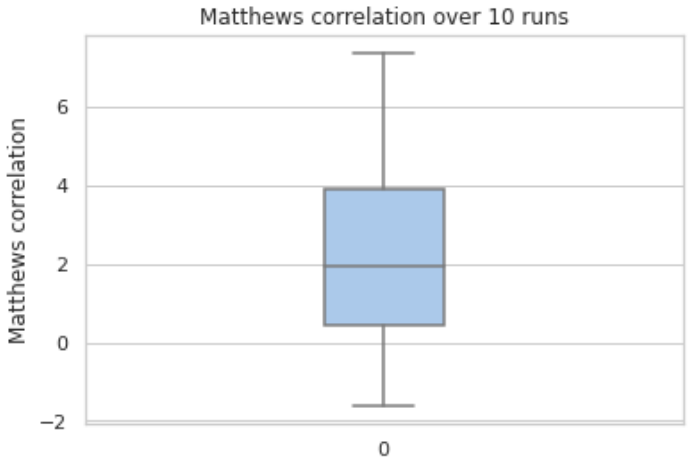


Figure 38. Matthews correlation over 10 runs for CoLA

The plot shows a relatively high variance, ranging from ~2 negative to ~6+ positive. It would be a good exercise to show which are the examples that the model predicts correctly when it does so. This would indicate which language signs are easier to catch by the model.

Seems intuitive that a more robust language model is needed in order to cope with this task. Future work in this dataset can include the BERT tokenizer and potentially a training setting where the triplet loss is calculated.

Triplet loss is known to be difficult to implement (Moindrot, 2018). It's a technique introduced in face recognition to train models in detecting similar faces and dismiss dissimilar ones. Similarly, this technique could try to maximize the differences in the representation of sentences that are ungrammatical and minimize the difference between those that are grammatical.

An experiment was conducted using the triplet loss off-the-shelf implementation from TensorFlow without positive results. The problem seems to be that for it to work correctly, it is crucial to prepare batches in the form of triplets. Triplets are formed by one anchor sample, one positive sample and one negative sample. More information about the triplet loss can be found in Schroff, Kalenichenko, & Philbin, 2015. Future work on this task would benefit from an implementation that takes these aspects in consideration.

4.3 Microsoft Research Paraphrase Corpus

The MRPC (Dolan & Brockett, 2005) is a dataset that contains circa 4 thousand pairs of sentences and annotations indicating whether they have a semantic equivalence, that is, whether the two sentences have the same meaning. The labels are balanced for both equivalent and non-equivalent. It was published in 2005 and is still relevant, which gives an idea of the importance of the task. Since there are two inputs, one per sentence, the Siamese LSTM will be implemented as described in section 3.4. In Figure 39 a non-equivalent pair of samples is shown.

```
[14] print(df_train['sentence1'][0])
     print(df_train['sentence2'][0])
     print(df_train['label'][0])
```

```
b'The identical rovers will act as robotic geologists , searching for evidence of past water .
b'The rovers act as robotic geologists , moving on six wheels .'
0
```

Figure 39. MRPC example

The Siamese model is trained and obtains 69.11% test accuracy. Results are shown in Table 6. In green the reported measure for the task.

MPRC				
	Training	**Validation**	**Test**	**Max Value**
Samples	2,934	733	408	-
Accuracy	72.49	70.44	69.11	73.10

Table 6. MPRC accuracy

The results, being positive, leave room for improvement. In the following plot is shown how the tendency indicates that perhaps better results can be obtained by training over more epochs. This option was discarded for consistency with the other tasks, although might yield better performance.

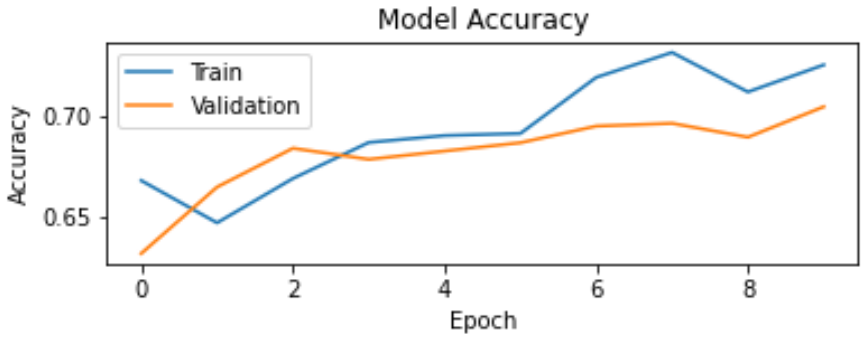


Figure 40. MRPC results with Siamese LSTM

To account for stochasticity, the experiments are run 10 times and average results presented. The median value is 68.25%, with a standard deviation of 0.0287, which shows that the model is stable and reliable.

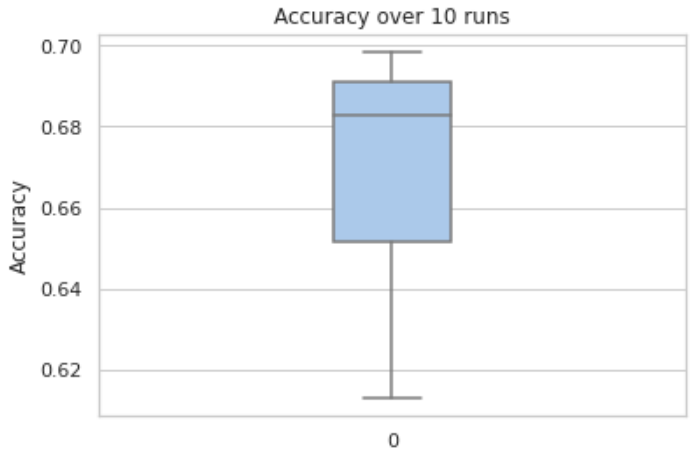


Figure 41. 10 runs accuracy

Better performance could be obtained by leaving the model in training mode for more epochs, which is discarded so the experiment is consistent with the rest of the tasks. Also, a more robust language model like BERT could be utilized and would potentially give better results. A training set that enables transfer learning will probably be beneficial since the available data is quite scarce for this task.

4.4 Quora Question Pairs

The QQP is a collection of sentences in the form of questions that are taken from the question-answering community Quora. It comes with annotations on whether the questions are duplicated or not. The following snippet gives an idea of how the dataset looks like

```
[ ] df_validation.tail()
```

	idx	label	question1	question2
40425	32434	0	b'Are both UY Scuti and VY Canis Majoris the s...	b'What would UY Scuti look like if it were the...
40426	26967	1	b'What would be his reaction if I give him a a...	b'What would be his reaction if I give him a l...
40427	18713	1	b'I recently found my dad is cheating on my mo...	b'My dad is cheating on my mom. What should I ...
40428	19274	1	b'Is Jesus historically real?'	b'Was Jesus real?'
40429	18576	0	b'Vaginaplasty: Who are the best surgeons in N...	b"How can you look at someone's private Instag...

Figure 42. Quora question pairs snippet

One of the challenges this dataset brings is the amount of data that has to be analysed, with many more samples than any other on the benchmark. The Siamese LSTM is trained in the dataset and yields and accuracy of 79.46% in test set. The overall F-Score is 77.90 and reports 83.78 precision for negative results and 72.02 for positive results. The results are shown in Table 7 and relevant metrics highlighted in green.

QQP				
	Training	**Validation**	**Test**	**Max Value**
Samples	291,079	72,769	40,430	-
Accuracy	79.71	80.07	79.46	80.07
F Score			77.90 (macro)	

Table 7. QQP results

The F-score has to be taken with caution as is just the off-the-shelf implementation from scikit-learn library and yields results that are somewhat unusual. It would be expected that the accuracy for negative values was lower than that of positives, since in general is more difficult to tell apart what is wrong than what is not. What is closer in the embedding space than what is further in this case.

The F-score in Scikit learn is defined as that which “Calculate metrics globally by counting the total true positives, false negatives and false positives.” The fact that it resembles so much to the accuracy gives ground to think there might be a mistake in the calculus. However, in the GLUE paper is reported that “it is observed a different label distribution than in the training set”, which might explain the similarity in the measure. This will imply that there is no class imbalance and thus the F-score would be the same as the accuracy.

The accuracy, in any case, is good overall and the model seem to converge at about 8 passes, although it might benefit from further training.

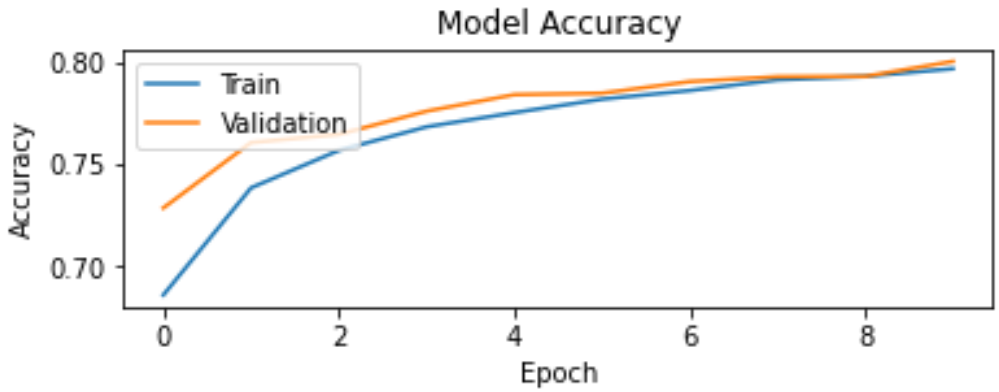


Figure 43. QQP accuracy after 10 epochs

It’s important to note that the difficulty of this task has to do with the amount of data, 57MB file and 400 thousand plus samples. The model takes about 8 minutes to complete a single pass when training with a TPU (Tensor Processing Unit) from Google. This makes every single experiment to last about 80 minutes and renders the idea of running 10 experiments out of reach since it would need 800 minutes, more than 12 hours of processing time, which is the limit set by Google Colab for using their facilities.

4.5 Semantic Textual Similarity Benchmark

The STS Benchmark (Cer et al., 2017) is a collection of paired sentences, each one with human annotations regarding its similarity. The similarity score ranges from 1 to 5, being 5 most similar and 1 most dissimilar. Figure 44 shows a few samples of the dataset.

```
[5] df_validation.head()
```

	idx	label	sentence1	sentence2
0	878	0.4	b"For what it's worth, dirt on the lens might ...	b'Regular dish washing fluid is great for remo...
1	787	3.4	b'GPS (wikipedia) is based on orbiting satelli...	b'GPS systems can, and do, work everywhere you...
2	458	2.6	b'a dog runs through the long grass.'	b'A dog lays down in the long green grass.'
3	1048	0.8	b"The broader Standard & Poor's 500 Index <.SP...	b'The technology-laced Nasdaq Composite Index ...
4	1112	2.2	b'Shares of LendingTree soared $5.99, or 40.1 ...	b'Shares of LendingTree rose $6.21, or 42 perc...

Figure 44. STS Snippet

Labels are converted to fall within 0 to 1 and then feed into the Siamese model. The model reports and accuracy of 09.07% in the test set. the following plot shows how the model evolves over different epochs.

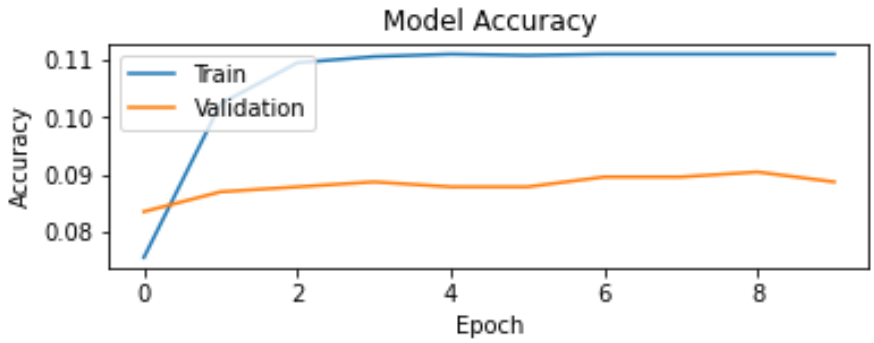


Figure 45 Accuracy for STSB.

Given the nature of the task, the accuracy does not provide and exact picture of how well the model performs. The labels are numbers within 0 and 1, and for a perfect accuracy, the model should match all predictions with the labels. Instead, correlation metrics are calculated that indicate the degree of adjustment between predictions and labels. The correlation results are 46.54 and 46.48 for Pearson and Spearman respectively over 100. Figure 45 shows the results in the experiments.

```
[46] from scipy.stats import pearsonr
     import scipy.stats

[47] correlacion_pearson = scipy.stats.pearsonr(x=Xs, y=gold)

[48] correlacion_pearson

     (0.46543374617121136, 1.6867116665454094e-81)

[49] correlacion_spearman = scipy.stats.spearmanr(a=Xs, b=gold)

[50] correlacion_spearman

     SpearmanrResult(correlation=0.4648049297209279, pvalue=2.9541171945793616e-81)
```

Figure 46. Pearson and Spearman correlation for STSB

These results, not being optimal, leave ground for further discussion. Table 7 illustrates them. Relevant results are in green.

STS-B				
	Training	**Validation**	**Test**	**Max Value**
Samples	4,599	1,150	1,500	-
Accuracy	11.11	08.87	09.07	11.11
Pearson/Spearman			46.54 / 46.48	

Table 8. Results for STSB

The experiments are run 10 times and the results plotted in Figure 47. The results are relatively inferior showing median values of 37.68 and 36.63 for Pearson and Spearman respectively. The standard deviation is 0.02475 and 0.02265, again very reliable figures.

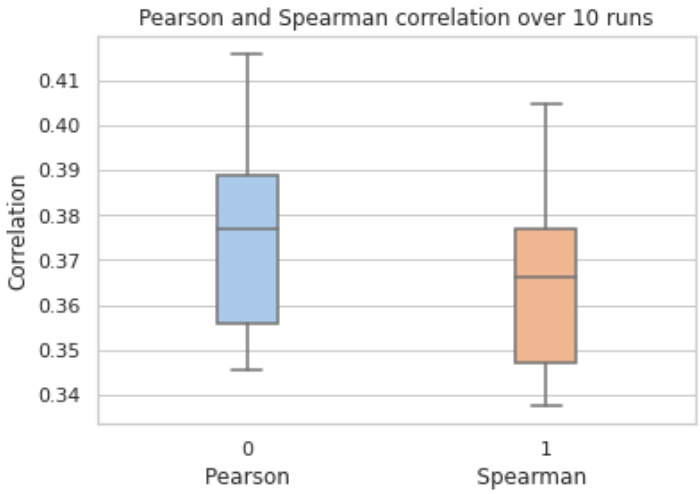


Figure 47. Pearson and Spearman Correlation STSB

The fact that the correlation metrics are lower than those related in single experiments is certainly a remarkable phenomenon, particularly when all conditions remain equal but one, the processing unit used. When running the experiments over 10 times, a TPU was used for speeding up the process. For the model to reset its weights in every experiment the following command is used:

```
tf.keras.backend.clear_session() # Starts session from scratch
```

The command cleans the backend and the graphs in TensorFlow. Perhaps this formula has to be different when using a TPU. This would make sense as the correlation declines in every experiment as shown in Figure 48, indicating that the experiments are related rather that independent of each other. At the moment this issue remains unsolved.

```
[SpearmanrResult(correlation=0.39556945483298184, pvalue=2.304826177336167e-57),
 SpearmanrResult(correlation=0.40453636316434277, pvalue=3.753819386813934e-60),
 SpearmanrResult(correlation=0.3775288213704707, pvalue=5.214027057283529e-52),
 SpearmanrResult(correlation=0.37552388582136004, pvalue=1.95758178288219e-51),
 SpearmanrResult(correlation=0.36143272661818304, pvalue=1.6485718549597876e-47),
 SpearmanrResult(correlation=0.3712440590205852, pvalue=3.196745262686123e-50),
 SpearmanrResult(correlation=0.3567061948824008, pvalue=3.0926089085499156e-46),
 SpearmanrResult(correlation=0.3440809156957049, pvalue=6.120781064885346e-43),
 SpearmanrResult(correlation=0.3375175858376866, pvalue=2.7638162887094403e-41),
 SpearmanrResult(correlation=0.34131516942604495, pvalue=3.0828497668329075e-42)]
```

Figure 48. Correlation declines with every new experiment

Furthermore, although this implementation follows closely the one from Mueller et al. (2016), the results are not as good as those they report. A partial explanation can be that they pretrained their LSTM on the STS dataset (in section 4.2).

Experiments conducted with 100 hidden dimensions in the LSTM instead of 50 show a slight increase in correlation, concretely: 48.87 and 47.89 for Pearson and Spearman. So nearly 2 and 1.5 percent increase respectively, however far from the results they report in a similar dataset, and also far from the results of other models reported by (Wang et al., 2019). The model is left with 50 neurons for consistency.

4.6 Results comparison and future work

This project started by asking how much attention mechanisms contribute to a model's performance in common NLP tasks. The model that was trained was an LSTM and a Siamese LSTM without attention mechanisms. This brings the opportunity to compare those results with the one's the Wang et al. (2019) report in the GLUE paper for their BiLSTM model with attention (discussed in section 2.3.2). The Bi in BiLSTM stands for bidirectional, a slightly refined version of the LSTM used in this project.

The following table shows the results for models with and without attention mechanisms, and the human performance on the same tasks. Similar to Nangia et al. (2019) the table reports the difference between human vs attention and attention vs non-attention models. Results can be seen in Table 9. In green column appears the differential, highlighted in red when this is negative.

	Single Sentence		**Similarity and Paraphrase**		
Model	**CoLa**	**SST-2**	**MRPC**	**QQP**	**STS-B**
Human performance	66.4	97.8	80.8	80.4/59.5	92.7/92.6
Δ Human - Att	50.7	11.9	12.3	-3.1	33.4/
BiLSTM + attention	15.7	85.9	68.5	83.5/62.9	59.3/55.8
Δ Att – non-att	12.7	4	-0.6	4.0/-15.0	12.8/9.3
Single LSTM/ Siamese LSTM	3.0	81.9			
			69.1	79.5/77.9	46.5/46.5

Table 9. Results for models with and without attention mechanisms and human performance

The results confirm that attention mechanisms contribute to the overall performance of the model, but perhaps not as much as would be expected. The Siamese LSTM manages to outperform by 0.6% in the MRPC task, although it falls short against the rest of the tasks.

The GLUE benchmark is fit to its purpose as it enables to see the performance of the models when pushed to their edges. For instance, it is interesting to note that relatively accessible tasks

as SST2, MPRC and QQP don't show major differences between models with and without attention, in fact non-attention models perform better in MPRC. Probably with subtler fine-tuning these differences could be reduced even more.

The reported F-Score for QQP that shows better performance for non-attention models have to be taken with caution as there's the possibility of the metric taken wrongly or the test set not being imbalanced in contrast with the training set. Section 4.4 explains this in more detail. It is also interesting to note how human performance is better in all tasks but in QQP, which the authors suspect might be due to the instructions of the task not being sufficiently concise (Nangia et al., 2019).

The greater differences can be observed in tasks where finer discernment is required, like CoLA and STSB. The differences are greater both between models with and without attention, but also between these and human performance. This seems to be pointing towards attention mechanisms being more able to show finer characteristics of human language than their competitors.

In any case, both models fall far behind human performance. It is assumed that human performance has been tested in English native speakers by Nangia (2019). It would be interesting to experiment with non-native speakers and observe the differences, since one of the constraints of the tasks is the relative scarcity of the training samples ~7k. It is observed how the model needed 20 epochs to converge in comparison with 10 for the rest of the tasks. The CoLA task is still the most challenging one for both humans and neural models.

STSB tries to adjust models' predictions with those made by humans, the fact that other humans are able to adjust these predictions better than neural models points out that there's something that scapes their capabilities. This is particularly remarkable for attention mechanisms, as they are designed precisely to calculate the probabilities of two given words being linked. The fact that they fall behind human capacities points there is room for improvement in this calculus.

Differences in performance between models could be also attributed to the architecture of the models, rather than solely the attention mechanisms. The authors' classifier has 512 dimensions

in the hidden layer whereas this project uses 50 following Mueller et al. (2016). Experiments with 512 D didn't show any major improvement in results.

Better performance in the tasks seems to happen when these are taken as multitask exercise, which allows for transfer learning across tasks. Transfer learning implies that the model incrementally 'learns' from one task to the next one, thus gaining more capabilities and performing better. This option also requires more complexity in the implementation and was discarded for this project, but future work should explore this direction.

The code for all tasks can be found at https://github.com/DiegoSnach/GLUE-Benchmark

5 Evaluation, future directions and conclusion

This chapter reflects on the work done for this project. It analyses it and compares it with the goals initially set. The challenges met during the process will be discussed, and also its weakest and strongest points.

The second part points out what would be the continuation of the project if more time would be granted. Then a final reflection is made on the work as a whole.

5.1 Evaluation

This project was set with the intention of using state of the art ML technologies and apply them to challenging NLP tasks. For doing so, the GLUE benchmark was chosen as a reference, for being a tool that aspires to develop models with understanding beyond detection of superficial correspondences between inputs and outputs (Wang, 2019).

While the idea of constructing a system able to process language in a generalized way is very appealing, it has proven to be excessively demanding given the specifications of this project.

The two main questions this project was set to explore are:

- How much attention mechanisms contribute to a model's performance?
- What is the trade-off between computing time and general efficiency?

The first question is answered in the project, showing that models with attention perform better in NLP tasks, although not as much as would be expected, being even outperformed by the Siamese LSTM in the MPRC task. The second question was partially answered since the model implemented was competitive compared to the attention one, but with a simpler architecture that enabled for low computational effort. Time constraints didn't allow further examination of the topic.

The first section started by reviewing the scientific literature and defining the topic at hand, namely, Natural Language Processing. It followed a description of different ML techniques that have been used for solving NLP tasks.

A historical review of ML approaches proceeded from the early developments of the discipline until current approaches. The reason for doing a somewhat extensive review is because is easy to fall under the impression that there's something mysterious about the behaviour of neural models, and to assume them as some sort of black boxes where the mechanisms that transform their inputs into outputs are obscure and elusive.

The historical review attempted to shed light on this issue and explain how the latest developments are often refinements of the previous ones. The more complex models of nowadays still rest in ideas surrounding RNNs and LSTMs, which were developed in 1990 and 1997 respectively, as shown in section 2.2.3.

The paper of Vaswani "Attention is all you need" of 2017 was a shift in the field with the introduction of models entirely based on attention. The original idea of this project was to explore attention mechanisms in greater detail. For that reason, solid baselines were needed to allow robust comparisons.

At the time of creating those baselines, time constraints and the complexity of the tasks made infeasible to produce both baselines and models with attention to test them. Luckily, the performance of attention mechanisms had been reported in the original paper, and this enabled comparisons between them and the developed baseline models.

The libraries needed for solving the tasks were also a source of complexity. It was discussed in section 3.1 the choice of framework, but what was assumed is they would have to run in TensorFlow, which is indeed the reference framework for the ML community. Although TensorFlow comes with all the needed libraries and makes the GLUE datasets readily available, there's a learning curve that has to be climbed when using it.

The development of these kind of projects is often exponential, which means that first steps require more effort and yield lower absolute progress. TensorFlow works with tensors, which

are common arrays of immutable nature, so to operate with them they have to be transformed. Lack in observance of this fact had the project stuck for a number of days and brought significant frustration. It was felt at moments that the documentation was scarce and not systematic in its approach, rendering it inaccessible for a relative newcomer.

When confronting the tasks in the benchmark, it was apparent that strong programming skills were needed to succeed, and that there was a significant jump from previous knowledge. Moreover, being the benchmark so recent, there weren't clear guidelines to follow, not many papers had discussed it, and not many implementations served as reference. A substantial amount of imagination and creativity was needed to adapt the model from other tasks and fit it into the required ones.

The mere fact of running the experiments from end to end is considered a major success. Indeed, once the general architecture is built, chances are that a model with attention could be ensembled without substantial difficulties. Time constraints and the possibility of ending up without any results made this option not sensible.

A tool that was instrumental for the development of this project was Colab from Google. It would have been much more complicated to run the experiments on a conventional CPU with the time restrictions imposed. The fact of having all major libraries readily available and the possibility of running GPUs and TPUs made it possible to experiment different architectures and parameters.

5.2 Future work and conclusions

Future work in this project would necessarily have to implement a model with attention. Two main approaches are available in the Keras library, Bahdanau and Luong style attention. Exploring those approaches and comparing them with the Transformer model from Vaswani would be a solid field for research.

If this project was to be made again, perhaps more weight would have been given to the implementation of the code, and less to the literature review. This reflects the domains were the author feels more comfortable with, although the whole project idea was to flip that balance towards the programming side.

All things considered, this project has succeeded even in its partial failures. The fact of confronting tasks of high complexity enables much more progress than that of resolving a simpler task to perfection.

Indeed, the GLUE benchmark was created with the aim of driving research towards more generalizable systems able to understand human language. Although this project is far from having achieved this goal, the developed skills make such scenario more likely to happen in the future.

6 References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). *NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE*. Retrieved from https://arxiv.org/pdf/1409.0473.pdf

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Bos, J. (2011). A Survey of Computational Semantics: Representation, Inference and Knowledge in Wide-Coverage Text Understanding. *Language and Linguistics Compass*, *5*(6), 336–366. https://doi.org/10.1111/j.1749-818x.2011.00284.x

Bringsjord, S., & Govindarajulu, N. S. (2020). Artificial Intelligence (E. N. Zalta, Ed.). Retrieved June 23, 2020, from Stanford Encyclopedia of Philosophy website: https://plato.stanford.edu/entries/artificial-intelligence/#ResuNeurTech

California State University Long Beach. (n.d.). History of the Perceptron. Retrieved from web.csulb.edu website: https://web.csulb.edu/~cwallis/artificialn/History.htm

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., … Kurzweil, R. (2018). Universal Sentence Encoder. *ArXiv:1803.11175 [Cs]*. Retrieved from https://arxiv.org/abs/1803.11175

Chollet, F. (2019). *On the Measure of Intelligence*. Retrieved from https://arxiv.org/pdf/1911.01547.pdf

Collobert, R., Weston, J., Com, J., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, *12*, 2493–2537. Retrieved from http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf

Dansbecker. (2018, May 7). Rectified Linear Units (ReLU) in Deep Learning. Retrieved November 27, 2019, from Kaggle.com website: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning

Eisenstein, J. (2018). *Natural Language Processing*. Retrieved from MIT Press website: http://cseweb.ucsd.edu/~nnakashole/teaching/eisenstein-nov18.pdf

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In International Conference on Artificial Intelligence and Statistics, pp. 315–323.

Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research*, *57*, 345–420. https://doi.org/10.1613/jair.4992

Graves, D., & Pedrycz, W. (2009). Fuzzy prediction architecture using recurrent neural networks. *Neurocomputing*, *72*(7–9), 1668–1678. https://doi.org/10.1016/j.neucom.2008.07.009

Harris, Z. S. (1954) Distributional Structure, *WORD*, 10:2-3, 146-162, DOI: 10.1080/00437956.1954.11659520

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014, June 1). A Convolutional Neural Network for Modelling Sentences. https://doi.org/10.3115/v1/P14-1062

Kim, Y. (2014). *Convolutional Neural Networks for Sentence Classification*. Retrieved from https://arxiv.org/pdf/1408.5882.pdf

Kostadinov, S. (2019, November 10). Understanding Encoder-Decoder Sequence to Sequence Model. Retrieved from Medium website: https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346

Lecun, Y., Denker, J. S., Solla, S. A., Howard, R. E., & Jackel, L. D. (1989). Optimal brain damage. *Advances in Neural Information Processing Systems (NIPS 1989), Denver, CO*. Retrieved from https://nyuscholars.nyu.edu/en/publications/optimal-brain-damage

Lipton, Z., Berkowitz, J., & Elkan, C. (2015). *A Critical Review of Recurrent Neural Networks for Sequence Learning*. Retrieved from https://arxiv.org/pdf/1506.00019.pdf

Maas, A. (2011). Sentiment Analysis. Retrieved July 30, 2020, from Stanford.edu website: http://ai.stanford.edu/%7Eamaas/data/sentiment/

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, *52*(1–2), 99–115. https://doi.org/10.1007/bf02459570

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Retrieved from arXiv.org website: https://arxiv.org/abs/1310.4546

Migdal, P., & Jakubanis, R. (2018, June 26). Keras or PyTorch as your first deep learning framework. Retrieved July 22, 2020, from deepsense.ai website: https://deepsense.ai/keras-or-pytorch/

Mitchell, T. M. (1997). *Machine learning*. Singapore: Mcgraw-Hill.

Nangia, N., & Bowman, S. (2019). *Human vs. Muppet: A Conservative Estimate of Human Performance on the GLUE Benchmark.* Retrieved from https://arxiv.org/pdf/1905.10425.pdf

Ng, A. (2019). Unsupervised Feature Learning and Deep Learning Tutorial. Retrieved from Stanford.edu website: http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

Olah, C. (2015). Understanding LSTM Networks -- colah's blog. Retrieved from Github.io website: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pennington, J., Socher, R., & Manning, C. (2014). *GloVe: Global Vectors for Word Representation*. Retrieved from https://nlp.stanford.edu/pubs/glove.pdf

Rosenblatt, F. (1962). *Principles of neurodynamics : perceptrons and the theory of brain mechanisms*. Washington D.C.: Spartan Books.

Rubenstein, H., & Goodenough, J. B. (1965). Contextual correlates of synonymy. *Communications of the ACM*, *8*(10), 627–633. https://doi.org/10.1145/365628.365657

Sahlgren, M. (2006). *The distributional hypothesis* *. Retrieved from http://soda.swedish-ict.se/3941/1/sahlgren.distr-hypo.pdf

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). *FaceNet: A Unified Embedding for Face Recognition and Clustering*. Retrieved from https://arxiv.org/pdf/1503.03832.pdf

Steedman, Mark, 2007, "Presidential Address to the 45th Annual Meeting of the ACL", Prague, June 2007. Also printed in 2008, "On becoming a discipline", Computational Linguistics, 34(1): 137–144. [Steedman 2007 [2008] available online]

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. Retrieved from arXiv.org website: https://arxiv.org/abs/1409.3215

Torfi, A., Shirvani, R., Keneshloo, Y., Tavaf, N., & Fox, E. (2020). *Natural Language Processing Advancements By Deep Learning: A Survey*. Retrieved from https://arxiv.org/pdf/2003.01200.pdf

Turc, I., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *ArXiv:1908.08962 [Cs]*. Retrieved from https://arxiv.org/abs/1908.08962

TURING, A. M. (1950). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, *LIX*(236), 433–460. https://doi.org/10.1093/mind/lix.236.433

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Brain, G., Research, G., … Polosukhin, I. (2017). *Attention Is All You Need*. Retrieved from https://arxiv.org/pdf/1706.03762.pdf

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019). *GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND-ING*. Retrieved from https://openreview.net/pdf?id=rJ4km2R5t7

Wang, X., Liu, Y., SUN, C., Wang, B., & Wang, X. (2015b). Predicting Polarities of Tweets by Composing Word Embeddings with Long Short-Term Memory. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the7th International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1343–1353, Beijing, China. Association for Computational Linguistics.

Wu, Y., Liu, L., Pu, C., Cao, W., Sahin, S., Wei, W., & Zhang, Q. (2019). A Comparative Measurement Study of Deep Learning as a Service Framework. *IEEE Transactions on Services Computing*, 1–1. https://doi.org/10.1109/tsc.2019.2928551

Appendix 1

This is where your project proposal goes

EDINBURGH NAPIER UNIVERSITY SCHOOL OF COMPUTING

MSc RESEARCH PROPOSAL

The process of completing and reviewing the contents of this form is intended ensure that the proposed project is viable. It is also intended to increase the chances of a good pass. Much of the material produced while completing this form may be reused in the dissertation itself.

1. **Student details**

First name	Diego
Last (family) name	R. Sánchez
Edinburgh Napier matriculation number	40430644

2. **Details of your programme of study**

MSc Programme title	MSc Computing
Year that you started your diploma modules	2019

3. **Project outline details**

Please suggest a title for your proposed project. If you have worked with a supervisor on this proposal, please provide the name. You are strongly advised to work with a member of staff when putting your proposal together.

Title of the proposed project	Paying attention at the GLUE benchmark
Is your project appropriate to your programme of study?	
Name of supervisor	Kevin Sim

4. **Brief description of the research area - background**

Please do not describe your project in this section. Instead, provide background information in the box below on the broad research area in which your project sits. You should write in narrative (not bullet points). The academic/theoretical basis of your description of the research area should be evident through the use of citations and references. Your description should be between half and one page in length.

Natural Language Processing (NLP) comprises a set of methods for making human language accessible to computers. It is focused on the design and analysis of computational algorithms and representations for processing human language.

The goal of NLP is to succeed in activities that require human language abilities like extracting information from texts, translating between languages, answering questions, holding a conversation, etc. (Eisenstein, 2018).

The representations used for performing those activities are based on the **distributional hypothesis,** which states that words that occur in the same contexts tend to have similar meanings (Harris, 1954).

Contemporary approaches to NLP rely heavily on Machine Learning (ML). For words to be processed by ML models, they need some sort of numeric representation. This representation takes the form of numeric vectors that capture the semantic relationships between words in a given corpus, based on the distributional hypothesis. Such representations are commonly known as word embeddings (Bengio et al., 2003).

Recurrent neural networks, Long Short-Term Memory (LSTM), and Gated Recurrent (GRU) neural networks in particular, have been firmly established as state-of-the-art approaches in NLP. The best performing models connect encoder and decoder mechanisms through **attention mechanisms** (Waswani, 2017). Attention mechanisms enable contextualized representations of words by calculating the probability of two given words being linked.

NLP aims at understanding human language and processing it in a way that is not exclusive to a single task. The General Language Understanding Evaluation benchmark (GLUE) is a collection of datasets used for training, evaluating and analysing NLP models relative to one another, with the goal of driving "research in the development of general and robust natural language understanding systems." (Wang, 2019).

The collection consists of nine "difficult and diverse" task datasets designed to test the language understanding of a model (McCormick, 2019). These tasks involve question answering, sentiment analysis and textual entailment.

The aim of this project is to design a model architecture capable of dealing with a set of these tasks. To achieve this, state-of-the-art technologies will be utilized, with a specific focus on lightweight implementations that enable low computational power (Turc et al., 2019).

5. **Project outline for the work that you propose to complete**

Please complete the project outline in the box below. You should use the emboldened text as a framework. Your project outline should be between half and one page in length.

The idea for this research stems from:

Having worked in a customer service office, I have realized how a great proportion of the day-to-day communication with customers can be easily automated with the aid of algorithms. The subjects I did during my masters have brought me closer to being able to develop and implement this idea myself.

The aims of the project are:

- To compare the performance of different NLP models with and without attention mechanisms in a set of tasks from the GLUE benchmark.
- To investigate the trade-off between computing effort and performance of the different NLP models in the GLUE benchmark

The main research questions that this work will address include:

- What is the trade-off between computing effort and performance in state-of-the-art NLP technologies?
- How much do attention mechanisms contribute to a model's performance in common NLP tasks?

The software development/design work/other deliverable of the project will be:

- An ML model implementation for a set of tasks proposed in the GLUE benchmark

The project deliverable will be evaluated as follows:

- Accuracy and F1, following the GLUE guidelines

The project will involve the following research/field work/experimentation/evaluation:

- A practical implementation of state-of-the-art models in the GLUE benchmark.

This work will require the use of specialist software:

This work will require the use of specialist hardware:

The project is being undertaken in collaboration with:

6. **References**

Please supply details of all the material that you have referenced in sections 4 and 5 above. You should include at least three references, and these should be to high quality sources such as refereed journal and conference papers, standards or white papers. Please ensure that you use a standardised referencing style for the presentation of your references, e.g. APA, as outlined in the yellow booklet available from the School of Computing office and http://www.soc.napier.ac.uk/~cs104/mscdiss/moodlemirror/d2/2005_hall_referencing.pdf .

Eisenstein, J. (2018). Natural Language Processing. [online] Available at: http://cseweb.ucsd.edu/~nnakashole/teaching/eisenstein-nov18.pdf [Accessed 28 May 2020].

Harris, Z. (1954). Distributional structure. Word, 10(23): 146-162.

Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., Jauvinc@iro Umontreal Ca, Kandola, J., Hofmann, T. and Poggio, T. (2003). A Neural Probabilistic Language Model. Journal of Machine Learning Research, [online] 3, pp.1137–1155. Available at: http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Brain, G., Research, G., Jones, L., Gomez, A., Kaiser, Ł. and Polosukhin, I. (2017). Attention Is All You Need. [online]

Available at: https://arxiv.org/pdf/1706.03762.pdf.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S. (n.d.). GLUE: A MULTI-TASK BENCHMARK AND ANALYSIS PLATFORM FOR NATURAL LANGUAGE UNDERSTAND- ING. [online] Available at: https://openreview.net/pdf?id=rJ4km2R5t7 [Accessed 26 Dec. 2019].

McCormick, C., 2019. https://mccormickml.com/2019/11/05/GLUE/. [Blog].

Turc, I., Chang, M.-W., Lee, K. and Toutanova, K. (2019). Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. arXiv:1908.08962 [cs]. [online] Available at: https://arxiv.org/abs/1908.08962 [Accessed 28 May 2020].

7. **Ethics**

If your research involves other people, privacy or controversial research there may be ethical issues to consider (please see the information on the module website). If the answer below is YES then you need to complete a research Ethics and Governance Approval form, available on the website: http://www.ethics.napier.ac.uk .

Does this project have any ethical or governance issues related to working with, studying or observing other people? (YES/NO)	

8. **Confidentiality**

If your research is being done in conjunction with an outside firm or organisation, there may be issues of confidentiality or intellectual property.

Does this project have any issues of confidentiality or intellectual property? (YES/NO)	

10. **Submitting your proposal**

1. Please save this file using your surname, e.g. macdonald_proposal.docx, and e-mail it to your supervisor, who will discuss it with you and suggest possible improvements.
2. When your supervisor is content with your proposal, submit it to the Research Proposal Upload link on Moodle, and email your internal examiner to notify them that you have submitted. They will leave feedback for you on Moodle.
3. Discuss your feedback from the internal examiner with your supervisor and if necessary make final changes to your proposal.
4. When you produce your dissertation, add your finalised proposal as an appendix.