Advancing duplicate question detection with deep learning

Pumnea, A.M.

Award date:
2019

Awarding institution:
Technische Universität Berlin

Link to publication

Technische Universität Berlin

Faculty of Electrical Engineering and Computer Science
Big Data Management Group

Master Thesis

Advancing Duplicate Question Detection with Deep Learning

Andrada Maria Pumnea

Matriculation Number: 396477

October, 2018

Supervised by
Prof. Dr. Ziawasch Abedjan

Second Reviewer
Prof. Dr. Klaus-Robert Müller

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 31.10.2018

. . . .

Andrada Maria Pumnea

Abstract

Identifying duplicate questions in online forums and communities is of utmost importance for ensuring a high-quality user experience. Natural language allows for many different ways to formulate the same question, which makes the task of duplicate question detection very challenging. Deep learning solutions have become the state-of-the-art, especially as larger datasets have become available for research.

The problem of identifying duplicate questions when data is scarce has been largely overlooked. The process of labeling data is expensive and time-consuming, as a high-quality dataset requires multiple experts to label the questions and vote on the ground truth. Hence, it is desired that the new techniques are able to learn as much as possible from smaller training sets and have a good performance on unseen pairs of questions. Most of the approaches proposed in literature rely on large amounts of labeled data to train their deep learning models, which is not likely to be possible in a real-life scenario.

In this thesis, we propose four components that aim to improve the classification results and conduct extensive experiments on three state-of-the-art models to understand their impact. The results show that by combining pre-trained word and sentence embeddings, we can already improve the performance of the classifiers. In addition, simple data augmentation techniques, such as reversing question order, can be effective as well. We show that by training the models on a related dataset and fine-tuning it on the smaller samples, sentence encoding models will benefit the most, in comparison to sentence interaction models. Finally, we attempt to create a hybrid solution by ensembling the deep learning models, choosing the best performing text representation and augmenting the dataset. While this solution outperforms all our baselines on all our reduced training samples where the pairs of questions are short and have a high word overlap, it is less effective on the dataset with long questions and little overlap.

Zusammenfassung

Die Identifizierung doppelter Fragen in Online-Foren und Communities ist von größter Bedeutung für die Sicherstellung einer qualitativ hochwertigen Benutzererfahrung. Die natürliche Sprache ermöglicht viele verschiedene Möglichkeiten, die gleiche Frage zu formulieren, was die Aufgabe der Erkennung doppelter Fragen sehr schwierig macht. Deep-Learning-Lösungen sind zum Stand der Technik geworden, zumal größere Datensätze für die Forschung verfügbar geworden sind.

Das Problem der Identifizierung doppelter Fragen bei knappen Daten wurde weitgehend übersehen. Der Prozess der Kennzeichnung von Daten ist teuer und zeitaufwendig, da ein hochwertiger Datensatz mehrere Experten erfordert, um die Fragen zu kennzeichnen und über die Grundwahrheit abzustimmen. Daher ist es wünschenswert, dass die neuen Techniken in der Lage sind, so viel wie möglich von kleineren Trainingsmengen zu lernen und eine gute Leistung bei zuvor nicht gesehenen Fragenpaaren zu erbringen. Die meisten der in der Literatur vorgeschlagenen Ansätze beruhen auf großen Mengen annotierter Daten, um ihre Modelle des tiefen Lernens zu trainieren, was in einem realen Szenario wahrscheinlich nicht möglich ist.

In dieser Arbeit schlagen wir vier Komponenten vor, die darauf abzielen, die Klassifikationsergebnisse zu verbessern und umfangreiche Experimente an drei modernen Modellen durchzuführen, um deren Auswirkungen zu verstehen. Die Ergebnisse zeigen, dass wir durch die Kombination von vortrainierten Wort- und Satzeinbettungen bereits die Leistung der Klassifikatoren verbessern können. Darüber hinaus können auch einfache Datenergänzungstechniken, wie z.B. die Umkehrung der Wortreihenfolge in einer Frage, effektiv sein. Wir zeigen, dass durch das Training der Modelle auf einen verwandten Datensatz und die Feinabstimmung auf die kleineren Stichproben die Satzkodierungsmodelle im Vergleich zu Satz- Interaktionsmodellen am meisten profitieren werden. Schließlich versuchen wir, eine hybride Lösung zu schaffen, indem wir die Deep-Learning-Modelle zusammenführen, die leistungsfähigste Textdarstellung auswählen und den Datensatz erweitern. Während diese Läsung alle unsere Baselines bei allen unseren reduzierten Trainingsstichproben übertrifft, bei denen die Fragenpaare kurz sind und sich hohe Wortüberschneidungen ergeben, ist sie im Datensatz mit langen Fragen und geringen Überschneidungen weniger effektiv.

Contents

List of Figures

List of Tables

1 Introduction

Natural Language Processing has witnessed a surge in advances in recent years, with researchers proposing impressive breakthroughs in the area of text classification. Text classification has attracted a lot of attention because of its immediate applicability to everyday life and products, changing the way people interact with technology. As machines become able to understand language, humans gain the advantage of being able to translate, extract information and detect duplicate texts faster.

The problem of detecting duplicates in the area of natural language processing is of utmost importance for online user forums. By online user forums we refer to question-and-answer (Q&A) websites, which in recent years have become increasingly popular, creating a safe space for Internet users to share their knowledge with other users. The reason why these websites are so appreciated is because all the actors involved in them have the opportunity to learn by having their questions answered or by engaging in an information exchange with other (possibly) expert users. Thus, it becomes critical that the content on these platforms is of high-quality and questions are answered in an efficient manner.

Detecting duplicate questions as soon as they are posted is a major task in the maintenance and curation of Q&A communities, such as Quora [1] and Stack Exchange [2]. These websites usually contain vast amounts of information and, more often than not, users will find themselves overwhelmed when having a particular query in mind. Having each question asked only once, first of all, brings the advantage of time efficiency for the users as they do not need to wait minutes or hours to have their questions answered; second, offers improved user experience through decluttering of the platform of redundant questions; finally, avoids a fragmented knowledge base by having all the answers collected under a single canonical question [Addair, 2017]. Hence, in case the same question has been asked several times before with a different phrasing, it is preferred to have these duplicate questions merged or removed.

Detecting duplicate questions is challenging, mainly because of the volatility of natural language and way the same meaning can be expressed with many different words. A system that is able to perform this task needs to thoroughly understand complex natural language issues such as semantics and grammar. There have been several valuable efforts towards accurately detecting duplicate questions. These efforts range from measuring the overlap of words between two questions to using machine learning and deep learning techniques that learn to differentiate between what is duplicate and what is non-duplicate.

[1]https://www.quora.com/
[2]https://stackexchange.com/

Most of the recent advancements use complex deep learning architectures as their main approach to detect duplicate questions and are based on the assumption that there is plentiful of labeled data available ([Wang et al., 2017], [Gong et al., 2017], [Kim et al., 2018]). There is a lack of understanding how deep learning approaches behave and how their performance varies in the use case of small labeled datasets. The main problem is that with limited datasets deep learning classifiers might not be able to generalize well on unseen data. We need to identify what text representation is suitable so that as much meaning as possible can be extracted from the data at hand and which techniques are appropriate for enhancing datasets or the performance of the classifiers in order to reduce overfitting.

This thesis proposes a solution which combines several components for text representation, data augmentation, transfer learning and ensembling. We study how state-of-the-art deep learning models perform in a setting where data is limited using two well known datasets and evaluate our proposed solution.

1.1 Purpose

This project analyzes the performance of state-of-the-art deep learning models in a setting where labeled data is scarce. It is expected to understand which strategies can be used to deal with the problem of small datasets and what is the best text representation that would contribute to correctly identifying duplicate questions. Specifically, in this thesis we aim to answer the following questions:

- *What is the behavior of deep learning models when they are dealing with limited datasets for duplicate question detection?*
- *What is the contribution of text representation, data augmentation, transfer learning and ensembling in correctly identifying duplicate questions when data is scarce?*

1.2 Problem

This thesis addresses the problem of duplicate question detection. Our purpose is to contribute to the ongoing development of techniques suitable for this task, by studying existing deep learning techniques and architectures and combining them to create an improved solution for the scenario when labeled training data is scarce. We use the same definition for duplicate questions, as the one provided by Bogdanova et al. who state that "two questions are semantically equivalent if they can adequately be answered by the exact same answer" [Bogdanova et al., 2015] .

The problem of identifying duplicate questions generalizes to a classification problem. The goal of **classification** is to map unseen inputs to one of the available classes. In our case, we need to determine whether a given pair of questions, is duplicate or non-duplicate. More formally:

$$f(q_1, q_2) \rightarrow 0/1$$

where f is a function learned by a model and 0 or 1 are the possible classes that we can assign the pair, where 0 is non-duplicate and 1 is duplicate. As there are only two possible classes or categories this is a binary classification problem, which can be solved through **supervised** means.

Paraphrasing is a complex language phenomena as it offers many ways to express the same meaning. Paraphrases can be identified on 24 different levels of granularity and they can be classified into four categories [Vila et al., 2014]. We discuss the most important ones, which we also encounter in our datasets and we use questions from Quora dataset [3] as reference:

1. **Morphology-based changes** refer to paraphrases that differ based on way words are formed. The discrepancy may occur on inflectional or derivational level. For example:

   a1. *What is the best story you have?*
   a2. *What are some of your best stories?*

   b1. What are some tips for plucking eyebrows?
   b2. How should I pluck my eyebrows ?

2. **Lexicon-based changes** are changes that occur at word level (spelling changes, synthetic substitutions and opposite-polarity substitutions). For example:

   c1.*How do I build an online directory ?*
   c2. *What is the best way to build an online directory?*

   d1. *Why is India not getting Olympic medals?*
   d2. *What is the valid reason for India only getting a few medals in the Olympics?*

3. **Structural-based changes** denote changes to the pattern of the sentence such as alternation. For example:

   e1.*How do I hack a mobile phone remotely?*
   e2.*How can I remotely hack a mobile phone?*

4. **Semantics-based changes** refer to sentences that use different words for the same meaning, like in the example below:

   f1.*How do I lose weight faster?*
   f2.*How do I reduce weight rapidly?*

   g1.*How I can improve my English communication?*
   g2.*How I can speak English fluently?*

As examples (a-g) show, paraphrases ca be obtained in many different ways and the boundaries between paraphrase categories are not that strict. When identifying whether

[3]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

a pair of questions are duplicates, we study whether or not they are asking for the same thing. Establishing the relationship between the two questions can be quite subjective and the absolute ground truth can be hard to find as that would require getting in touch with the original author of the question and confirming their intention. Thus, we can assume that datasets available for the study of duplicate question detection contain a reasonable amount of unintentional noise. This is also one of the reasons why creating and manually labeling a high-quality dataset is very expensive: it would require a few experts to separately read and label each question and then find a method to vote and agree on a final label.

Recent work on detecting duplicates has focused on developing sophisticated deep learning architectures (see Chapter 3) and training them on large datasets in order to achieve good performance. The use case where labeled data is not always abundant has been neglected, with most researchers assuming large training datasets for developing their techniques.

To the best of our knowledge there is not much research that analyzes how deep learning models would perform in a setting with scarce data for duplicate question detection. Examples (a-g) show just a few of the different ways to paraphrase questions, which further builds our case for models that should be able to generalize well from seeing just a few examples from the paraphrase spectrum. Furthermore, there is no clear understanding regarding the current performance of state-of-the-art models with limited datasets and how they would improve with the help of different techniques for enhancing existing data and classifiers.

1.3 Contributions

This thesis describes an approach to combine different techniques for improving the classification results for duplicate questions in a setting with limited data. We conduct extensive experiments on four components that we propose for mitigating the challenge of scarce training data. Specifically, we analyze the impact of using different text representation, data augmentation, transfer learning and ensembling techniques. In addition, we create a hybrid solution where we ensemble our best performing deep learning models trained with the most promising text representation and data augmentation techniques. While this solution improves the baselines by a small margin on one of our datasets, the results suggest that the approaches need more data and better labels in order to improve significantly.

1.4 Outline

In **Chapter 2**, we introduce relevant theory and background in the area of deep learning and natural language processing. In addition, we present neural networks architectures typically used for natural language processing and techniques for obtaining word embeddings. **Chapter 3** introduces related deep learning models commonly used for identifying duplicate questions. Furthermore, we present three similar natural language processing problems and the field of duplicate detection in general. In **Chapter 4** we explain each component of the solution. We detail the different strategies employed for text representation, data augmentation, transfer learning and ensembling. We provide in **Chapter 5** an overview of the datasets used and discuss implementation details of the experimental framework. Furthermore, we analyze the results obtained from the experimental phase and offer an extensive discussion on the assumptions made. Finally, in **Chapter 6** we summarize our work and draw the conclusions. Furthermore, we introduce ideas for future work.

2 Theoretical Background

Researchers have been working on solving the problem of identifying duplicate pairs through numerous methods, ranging from rule-based techniques to heavy feature engineering and machine learning. Considering the recent advances in Deep Learning(DL), especially in the area of natural language processing(NLP), we propose a solution that leverages the newer technologies in this field. The aim of this section is to provide the theoretical context behind DL, focusing on its application for NLP. First, we introduce terms and concepts related to DL, together with the intuition behind them. Second, we introduce architectures which are used both by state-of-the-art approaches and our work: recurrent neural networks, long-short term memory networks, gated recurrent unit networks, bidirectional and siamese networks and attention mechanism. Finally we present the concept of word embeddings and describe algorithms that are related to our work such as GloVe [Pennington et al., 2014], FastText [Bojanowski et al., 2016] and Universal Sentence Encoder [Cer et al., 2018].

2.1 Introduction to Deep Learning

This section briefly introduces the concept of Deep Learning and motivates why we chose techniques from this field for performing Duplicate Question Detection (DQD).

Machine Learning (ML) is a broad field that encompasses techniques and approaches used to extract patterns from the data given as input. The representation of data fed into a ML algorithm plays a major role, as it affects the algorithm's ability to efficiently extract signals and make decisions. Thus, it is important to carefully select the information included in such a representation. Formally, the representation is composed of multiple features extracted from raw data. The process of creating new features requires good and intuitive understanding of the data at hand, becoming incrementally time-consuming with the sophistication of the new features. Thus, the biggest challenge of handcrafted features is deciding which features are important and relevant to the problem [Goodfellow et al., 2016].

Representation learning undertakes the task of discovering a useful set of features by means of ML techniques. Basically, it aims to make this process automatic [Goodfellow et al., 2016]. Thus, it cuts some of the human effort and time required to engineer features for complicated problems, ensuring improved performance. As Goodfellow et al. explain in their book, **Deep Learning (DL)** solves the main problem of representation learning, which is "extracting high-level, abstract features". Specifically, DL algorithms extend machine learning approaches by learning to represent more sophisticated concepts based on simpler ones (e.g. learn to represent an image of a house by first representing corners and contours). As Goodfellow et al. explain:

'Deep learning is a particular kind of machine learning that achieves great power and flexibility by representing the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.'

The recent success of DL algorithms is mainly due to the better performing computer hardware, which enables the efficient processing of the data. What is more, DL techniques have been delivering promising results on unstructured data such as text. Traditionally, most solutions to NLP problems consist of carefully handcrafted features and representations. This approach is cost-intensive both effort and time wise, while the feature set is still possibly incomplete. This is where DL comes into play, taking over the task of feature engineering. Modeling the relationship of a pair of questions and identifying whether or not they are duplicates, is inherently a complex NLP problem which requires a model to have a good understanding of syntax and semantics. These reasons indicate DL techniques as an efficient and effective solution to tackling the DQD problem. The rest of this thesis is concerned with leveraging the ability of DL to learn from text data in order to advance the study of DQD.

2.2 Neural Networks

This section introduces the main concepts related to neural networks. Neural networks have been around since the 1940s and could initially handle only one hidden layer. But with the development of technologies and hardware it became possible to build deeper, more effective architectures, which leads to deep learning as we know it today. This makes the study of neural networks in our context an interesting starting point, before diving deeper into concepts related to NLP.

2.2.1 Brief History

Initially, neural networks were inspired by how the biological brain works, which is why deep learning was also called **artificial neural networks (ANNs)** [Goodfellow et al., 2016]. In biology, a neuron is the cell that receives, processes and transmits information to other neurons through connections called synapses [neu, 2018]. On the other hand, artificial neurons are defined as computational units (usually mathematical functions) that take one or more inputs and generate an output.

McCulloch and Pits designed an initial version of the neuron as a linear model in 1943, aiming to replicate brain function [McCulloch and Pitts, 1943]:

$$f(x, w) = x_1w_1 + x_2w_2 + ... + x_nw_n$$

where $x_1, ..., x_n$ are the input values and $w_1, ..., w_2$ is a set of hand-chosen weights. Based on the linear model, a label y would be assigned. Inspired by their work, Rosenblatt created the *perceptron* which became the first model able to learn the set of weights

$w_1, ..., w_n$ [Rosenblatt, 1958]. The perceptron gave promising results, but could only classify linear patterns, which is a major shortcoming when dealing with real-world data. A few years later the **multilayer perceptron (MLP) or feedforward network** [Goodfellow et al., 2016] came as a breakthrough, as it became obvious that two or more layers could capture more complicated patterns than a single layer perceptron. A feedforward network is modeled as a directed acyclic graph, where each neuron in a layer has directed connections to the neurons in the following layer. Networks may contain feedback loops, which feed the outputs back into the model. In this case, they are called **recurrent neural networks** and they perform highly on NLP problems. These type of networks and techniques associated with them will be our main focus as they are able to deal well with long sequences of text data. The reason behind their promising results for text-based data is further explained in Section 2.3.1.

2.2.2 Components of an artificial neuron

A simple artificial neural network (ANN) consists of input layer, hidden layer and output layer, where the values of the hidden layer are used as inputs for the output layer. A network with several layers is known as a *deep* neural network. Data flows through the **neurons** of the layers with each neuron transforming the input it receives and forwarding it to the next layer. The neurons share the same characteristics irrespective of the layer they are part of.

The main components of an artificial neuron include inputs, weights, activation function and output(s). On the high-level, the inputs are multiplied by weights, then an activation function is applied to the result and finally, another function computes the output. The components of an artificial neuron are [Bangal, 2009]:

- **Weights** are defined as adaptive coefficients, whose values are changed during the learning process. Each input of a neuron is multiplied by a relative weighting factor, which decides the impact it will have further in the computation.
- **Summation function** helps combine the input and weights in different ways, before passing the result to the activation function. Denote the input as $x = [x_1, x_2, ...x_n]$ and weight vector as $W = [w_1, w_2, ...w_n]$. Thus, the summation function could be defined as the dot product between these two vectors:

  $$x^T W = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n$$

  In addition, the summation function could instead compute the minimum, maximum, product etc. depending on the designated network architecture. To generalize, the simplest form of an artificial neuron is a linear function which computes the weighted sum of inputs, to which, optionally, bias can be added:

  $$y = \sum x_i \cdot w_i + b$$

- The **activation function** transforms the result of the summation function (usually) in a non-linear way. If the function is linear, it would simply make the output proportional to the input. We define the activation function as g:

  $$y = g(\sum x_i \cdot w_i + b)$$

  The most common non-linear functions used as activation functions include:

  $$\begin{aligned} \text{sigmoid function} : \sigma(x) &= \frac{1}{1+e^{-x}} \\ \text{rectified linear unit (ReLU)} : ReLU(x) &= max(x, 0) \\ \text{hyperbolic tangent} : tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \end{aligned}$$

- The **output** is usually the result of an activation function.

2.2.3 Gradient based learning

This section introduces important concepts for designing and efficiently using neural networks, such as loss function and gradient descent. As discussed in the previous section, the weight W and bias b vectors are parameters to be learned on the training set. The purpose of a loss (sometimes called objective or cost) function is to help with learning these parameters. A **loss function** assesses the inconsistency between the predicted labels(ŷ) and the ground truth labels (y), thus computing the error over the training set. We aim to minimize the error as a function of weights and biases. This can be defined as an optimization problem, usually solved via gradient based-methods (**gradient descent or gradient ascent**, depending on whether we are looking for local minimum or maximum).

We define $L(\hat{y}, y)$ as a loss function, which associates a positive real number to the pair of predicted label $\hat{y}$ and true label y. This real number measures the error that the network makes on the data. Ideally the loss is 0 as we want $\hat{y} = y$. There are different ways to compute the loss of a neural network and each method will output a different value for the same prediction, thus having an impact on the performance of the model. We evaluate our deep learning models on the basis of the cross-entropy loss function:

- **cross-entropy** - shows the difference between the original distribution and the output distribution given by the model, where y_i is the ground truth and $\hat{y}_i$ is the actual prediction:

  $$L(\hat{y}, y) = -\sum_i y_i \log \hat{y}_i$$

Mathematically speaking, the **gradient** of the loss function is the partial derivative with respect to its parameters, measuring how much the output of a function changes if the parameters are varied [LeCun et al., 1998]. When we train a model via gradient descent, the trainable parameters w are updated through the following operation:

$$w_i = w_i - \lambda \frac{\partial L(w)}{\partial w_i}$$

where $L(w)$ is the loss function, λ is a scalar constant in the simplest case (also known as the **learning rate**). The learning rate determines how big the steps that the gradient takes in the direction of the minimum will be. This will be reflected in how fast or slow the gradient moves towards the optimal weights. It is important that the value of this parameter is set correctly, as a too high value runs the risk of missing the optimum, while a too small one runs the risk of getting stuck in a local minimum. The value for learning rate can be manually tuned by trying out different values, before satisfactory results are reached.

In order to adjust trainable parameters the update described above is performed as part of an algorithm called **backpropagation** [Rumelhart et al., 1986]. The algorithm has two phases, namely propagation and weight update: a forward pass is performed on the data and the loss is computed for the predicted labels; finally the errors are backpropagated layer by layer backwards and the parameters are modified accordingly.

Performing **Gradient Descent** (also known as **Batch Gradient Descent**) on very big datasets can be quite costly as the model weights are updated only after all training examples have been evaluated, in one big step. In practice, **Stochastic Gradient Descent (SGD)** is preferred as it entails calculating weight updates on subsets of training data, called *mini-batches*. The number of examples in a mini-batch is a hyperparameter to be determined based on experimental results. As setting the learning rate can be problematic, SGD was extended to use adaptive learning rates, where the learning rate is higher in the first iterations to ensure larger changes, only fine-tuning the weights in the end. Such techniques include Momentum [Qian, 1999], AdaGrad [Duchi et al., 2011], Adam [Kinga and Adam, 2015] etc. The choice of optimizer depends on the dataset at hand and it is usually decided experimentally.

Hyperparameter tuning is a time consuming process, as many different combinations of parameters are typically tried through Grid Search, before finally picking the best options. For the models included in this work, we mainly pick the parameters specified in the original paper. Further details are provided in Section 5.2.2 on how we choose these values.

2.2.4 Overfitting

Neural networks are able to learn complicated non-linear functions to fit any training set. On the downside, this may lead to overfitting where the neural network learns the training data so well that it is unable to generalize on new, unseen data. This problem can especially occur on datasets with a small amount of data to learn from.

In order to prevent the model from overfitting, it is recommended to evaluate it on a separate development set and stop the training once the model stops improving. Typically, if the loss is not decreasing, or on the contrary, it started increasing, then it means that the training needs to be stopped. This can be done through a technique called Early Stopping [Finnoff et al., 1993], which we also apply in our work.

Common techniques to stop overfiting include generating more data (i.e. data augmentation), regularization or Dropout [Srivastava et al., 2014]. Data augmentation entails finding appropriate ways to create new data that would help the classifier generalize better. We propose and evaluate as part of our solution some techniques to be applied to duplicate question detection which we describe in Section 4.3. Dropout deletes a number of units from the activations it is applied to, forcing the neural network to adapt. It bears the risk of losing some information and it is normally applied only during training. As for regularization, L1 and L2 regularization are commonly used for preventing the learnable weights to grow too much. These last two techniques are also used in our architectures, according to the original papers or tuning to the datasets.

2.3 Deep Learning for Natural Language processing

In this section we will review the main types of neural networks that will be used and discussed in this thesis. Text data is a special case of unstructured data and requires DL architectures that can deal with its peculiarities, in order to obtain good results. The main issues tackled by the models discussed in the next sections are the variability of text sequences and the understanding of context surrounding words. For DQD, understanding context is of utmost importance such that subtle differences and similarities between pairs of questions can be perceived by a DL model. The length of questions is also important as it is natural for users to express their queries with varying amounts of words and details.

2.3.1 Recurrent Neural Networks

Sequential data (such as text, speech, video) has complex temporal dependencies and a lot of hidden information. For example, the sentence "The cat has four ..." has a missing word, which humans can easily infer from the context. However, it can pose a tough challenge for a model to correctly guess that the word is "legs", unless it can take into consideration recent information (such as "cat") and connect it to the current context ("four").

Recurrent Neural Networks (or RNNs) [Rumelhart et al., 1986] were created in the 1980s to tackle these kind of problems, but like much of the deep learning models, they didn't get traction until recent years because of hardware capabilities. They are a group of neural networks that specialize in processing sequences of values $x_1, x_2, ..., x_n$. RNNs are able to scale to longer sequences and handle sequences of variable length [Goodfellow et al., 2016]. For example, after processing the words from the example above, a RNN would be able to understand that "legs" would follow in the context of "cat" and "four".

In feedforward networks (see Section 2.2.1), information can flow only forward i.e. there are no cycles or feedback loops. In contrast, RNNs have loops and allow for the information from the past to be used in the future. This is important in NLP problems, as the meaning of words is affected by previous words. For example, the meaning of "book" changes drastically from the context of "read a book" to "book a hotel". Thus, we need to be able to summarize the past and use it as input in the current state of the model.

Figure 2.1 depicts a visualization of the way RNNs work. In the first instance, an input x_t is passed through a cell A and outputs a value h_t. The loop allows the information to persist and to be passed from one state to the next one. In this cycle, every new input is concatenated to the output of the previous step, before the network can process it. The loop in the RNN can be represented by multiple copies of the same network, an idea which is called *unrolling*. Hence, the RNN becomes a multilayer network where parameters are shared across layers.

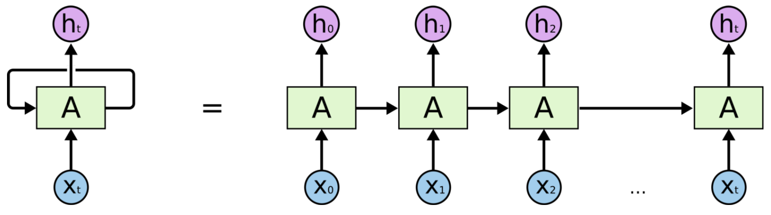


Figure 2.1: Unrolled recurrent neural network. Source: [Olah, 2015]

RNNs function on the basis of a chain reaction. All previous hidden states contribute to computing the current-time step h_t and this continues until all inputs of a sequence have been processed. Mathematically, we can formulate this as a recurrence equation, where we assume we have input x at time step t, and in order to compute the state h_t we need to take into account what happened at time step $t-1$ with state h. This renders the following recurrent equation, where θ are the trainable parameters:

$$h_t = f(h_{t-1}, x_t; \theta)$$

In practice, RNNs struggles with long-term dependencies, especially as the gap between words grows. Even though RNNs might be able to understand that "legs" follow "A cat has four..." as in our first example, it will struggle with long-term dependencies. For example, if we have a sentence that says "I live in Berlin [...] This city is the capital of ..." in which the recent context talks about a country but is not explanatory enough, RNNs would not be able to connect it to the context from further back. This is mainly because this type of networks involves composition on the same function multiple times, which can result in non-linear behavior. Thus, when doing **backpropagation through time** (essentially backpropagation on an unrolled RNN) the weights are multiplied many times and can end up vanishing (they become really small) or exploding (they become really big).

The solution to the **vanishing gradient** led to the introduction of the gated mechanism in the *Long Short Term Memory networks.* Although RNNs constitute the starting point in processing text data, this new type of networks are much more suited for the task of measuring text similarity and identifying duplicate questions, as we discuss in the next Section.

2.3.2 Long Short Term Memory Networks

Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] networks are a variation of recurrent neural networks that are able to deal well with long-term dependencies. Simply put, they are designed to be able to identify that the word is "Germany" in the previous example ("I live in Berlin [...] This city is the capital of ..."). The authors introduced in their paper an innovative *gating mechanism* that ensures a constant error flow through the special internal units, thus avoiding the long-term dependency problem. LTSM networks are explicitly designed to remember information for longer sequences and so they excel in dealing with various NLP problems, where the sequences can become quite lengthy and the meaning highly depends on the context (like in the case of DQD).

The key idea behind LSTMs is the *memory* which is stored in an internal state. There are three gates that constitute the gating mechanism: input gate, forget gate and output gate; their role is to decide which information should be included, added or removed from the memory. Simply put, a gate is made of a sigmoid layer and element-wise multiplication. It is in charge of deciding how much information can pass through it, regulating the amount of data. If the gate is closed (sigmoid returns a value close to zero) irrelevant inputs and noise do not enter the cell. The memory cell learns over time which information is important based on the weights it learns during the training procedure.

Each LSTM cell has a memory cell, an input gate, a forget gate and an output gate. All gates have the same dimension (the dimension of the hidden state) but each regulates a different kind of data:

- the **input gate** controls the newly computed state.
- the **forget gate** controls the previous state.
- the **output gate** controls the exposure of the internal state to the external network (higher layers).

For identifying duplicate questions, LSTMs are typically used to map the pair of questions to the same vector space given by the number of hidden units. This transformation is also known as *encoding* [Kalchbrenner and Blunsom, 2013]. The simplest way to proceed from here is to compute a similarity metric between the pair of questions and their new representations. Usually, more complicated transformations are applied to the new vector representations, as we will discuss in Chapter 3.

LSTMs have been successfully used in a number of different tasks including: handwriting recognition [Graves et al., 2009], speech recognition [Graves et al., 2013], machine translation [Luong et al., 2014]. In the area of DQD, several models that achieve state-of-the-art use LSTM to encode sentences ([Kim et al., 2018], [Gong et al., 2017], [Chen et al., 2016]). These models are discussed in 3.1.

2.3.3 Gated Recurrent Unit

Cho et al. introduced a simpler variant of the LSTM in 2014 [Cho et al., 2014]. This new type of network is called **Gated Recurrent Unit (GRU)** and proposes a gating mechanism similar to LSTM, but instead of three gates it requires just an *update gate* and a *reset gate*. The reset gate is trained to ignore irrelevant information and thus decides what to forget from the past. This means that when the gate is closed i.e. sigmoid is close to 0, the hidden state is reset with the current input only, effectively disregarding the previous hidden state. The update gate decides how much of the past information is passed to the future.

As LSTM and GRU are based on similar ideas, it is interesting to compare them and understand their differences. Chung et al. perform a thorough evaluation in their paper on the tasks of polyphonic music modeling and speech signal modeling [Chung et al., 2014]. From their experiments, it was clear that both the LSTM and GRU units outperformed the more traditional RNN. In terms of performance, it was not clear whether GRU was better than LSTM or the other way around, as they both produced similar results. In terms of efficiency, GRU trained faster and used less resources. On the other hand, Jozefowicz et. al. found that by adding a bias of 1 to the forget gate of the LSTM, results in better performance over GRU for tasks like arithmetic computation, XML modeling and language modeling [Jozefowicz et al., 2015].

GRUs have shown promising results in machine translation ([Firat et al., 2016], [Chung et al., 2016]). Similarly to LSTM, GRU is used to encode sentences. In Section 3.1.1, one of the models discussed uses GRU in their neural network architecture and obtains promising results. We also included this model in our experimental framework, which is presented in Chapter 5.

2.3.4 Bidirectional recurrent neural networks architectures

In this section, we will briefly introduce a variation brought to recurrent neural networks (and not only) in order to enhance the information it extracts. This approach is often seen in the models tackling pairs of sentences, like in the case of DQD.

We introduce the concept of **bidirectional RNN** [Schuster and Paliwal, 1997]. The typical *unidirectional RNN* has one hidden layer, as illustrated in Section 2.3.1. When applying it on text data, it can either be run from left to right (order of tokens would be $a, b, c, ...$) or right to left (order of tokens would be $c, b, a...$). Thus, it can only preserve information from the past or the future, i.e. the tokens it has already seen. On the other hand, a *bidirectional RNN* adds another hidden layer. Now the model can preserve information from both the past and the future, extending the properties of a simple RNN.

A bidirectional RNN consists of a forward pass and a backward pass, with the outputs from the two different passes not connected to each other. The forward pass (from left to right) will produce the hidden states $(\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_T)$, while the backward pass (from right to left) will encode the reverse sequence as $(\overleftarrow{h}_1, \overleftarrow{h}_2, ..., \overleftarrow{h}_T)$. The resulting hidden state will be of the form $h_i = [\overleftarrow{h}_i^T \overrightarrow{h}_i^T]^T$, as the outputs are combined through concatenation. What is more, any type of recurrent cell (RNN, LSTM or GRU) can be used for processing the input data in a bidirectional manner. In Section 3.1.2 we present some models for which this bidirectional encoding is essential for their performance. Furthermore, one of the models included in our framework uses BiLSTM to encode the pairs of questions. We discuss this model in Section 3.1.2 and Chapter 5.

2.3.5 Siamese Neural Network

The Siamese Neural Network architecture was proposed by Bromley et al. [Bromley et al., 1994] for solving the signature verification task and has been proven very successful in tackling the modeling of sentence pairs. It consists of two identical subnetworks (i.e. they have the same configuration in terms of weights, parameters etc.) each receiving a different input. Hence, each input is separately encoded and if they are of the same type (both text for examples) the resulting vectors are easier to compare or further process. In most cases, the results generated by the subnetworks are then used together by another block, which produces the final output. They have been proven efficient in identifying relationships between inputs, especially in problems like paraphrase identification (which is further discussed in Section 3.1.2) and question answering [Yin et al., 2015]. We included a model in our framework which uses this type of architecture for finding similarity between a pair of questions. This model is introduced in Section 3.1.1, with implementation details discussed in Chapter 5.

2.3.6 Attention mechanism

In this section we introduce the concept of *attention* [Bahdanau et al., 2014] which plays an important role in identifying the relationship of a pair of sentences and is widely used for tasks related to textual similarity, including DQD. This technique was created as an extension to the encoder-decoder architecture [Kalchbrenner and Blunsom, 2013], which struggled with squeezing long sequences of text into a fixed-length vector.

Briefly, the encoder-decoder architecture consists of a pair of neural networks, an encoder and a decoder, combined to work as a whole. In a typical setting, the encoder is responsible for mapping the input sequence to a vector with fixed dimensionality, while the decoder extracts the output sequence from this encoding.

The intuition behind the **attention mechanism** is that one fixed-length context vector is not sophisticated enough for storing all the information of a (long) sentence. Accordingly, multiple context vectors are computed, one for each word of the sentence, each containing information about the whole sentence but with a focus on the words surrounding the target word. This allows for a richer encoding. When decoding the sentence, the model now "chooses adaptively" the vectors to use by *soft-searching* significant information.

When we talk about **hard alignment** we refer to the more traditional alignment model in which one word from a source sentence precisely corresponds to a word from the target sentence. In contrast, **soft alignment** finds the link between any words from the source and target sentence which have a semantic or syntactic relationship with each other. For example, in a hard-alignment setting the model would "pay attention to the first word", while in soft-alignment setting it would "pay attention 60% to the first word, 20% to the second, 20% to the third".

Attention can be used in many different tasks. For example, it has shown promising results for image captioning [Vinyals et al., 2015], question answering [Sukhbaatar et al., 2015] and sentiment analysis [Kumar et al., 2016]. Two of the deep learning models that we use in our experimental framework efficiently use this mechanism to detect duplicate questions. These models are first introduced in Section 3.1.2 and further discussed in Chapter 5.

2.4 Word Embeddings

In this section we describe the role of word embeddings in dealing with text data. We cover how algorithms like Word2Vec [Mikolov et al., 2013b], GloVe [Pennington et al., 2014], FastText[Bojanowski et al., 2016] and Universal Sentence Encoder [Cer et al., 2018] transform words and characters to vector representation. This is necessary because machine learning models, including the neural networks in our framework, cannot process raw, unstructured text data.

2.4.1 Encoding textual data

Text data is typically raw and unstructured and needs to be processed before being fed to a classifier. There are two main ways to achieve this, specifically **one-hot encoding** and **dense embedding vectors** [Goldberg, 2017]:

1. **One-hot vectors** is a method for encoding categorical data (such as words) in a vector space model (for example bag-of-words) and it involves adding a new dimension to this new vector for each feature. To be more specific, each unique word in the vocabulary can be considered a feature and its presence will be marked by 1 or absence will be marked by 0. For example, assume we have a document with a vocabulary of 50,000 distinct words. When we use one-hot encoding on a sentence, each word in a sentence will be represented by a 50,000 dimensional vector where the position that corresponds to the word is marked with 1 and the rest are 0.

2. **Dense embedding vectors** represent words as real numbers in a lower dimensional space. For example, instead of representing a word as a $50,000$ dimensional vector, it will instead be represented by a 100 or 200 dimensional vector of real

numbers. Thus, dense embedding vectors are learned representations, where words which have the same meaning should have a similar representation. Moving from sparse to dense features, has had major computational benefits, making processing data and training DL models with text data much faster.

The task of DQD deals with textual datasets of reasonable size, where the context in which words appear and their meaning is important. For this reason we opt for using dense embedding vectors over one-hot vectors. There are multiple ways of generating these embedding vectors and we discuss these different methods in the following sections.

2.4.2 Using pre-trained embeddings

This section presents how the idea of using pre-trained embeddings has become widespread and we discuss the most important characteristics of embeddings that were conceived in this journey, focusing on *word2vec*. After the introduction of this technique, the field of word embeddings truly boomed, leading to the creation of the word embeddings that we use in this thesis.

Bengio et al. introduced word embeddings in 2003 [Bengio et al., 2003]. Their initial approach was to learn word vector representations as part of the architecture of a language model. Their model consists of an **embedding layer**, multiples intermediate layers and one softmax layer. By learning distributed word representation and the probability of word sequences from these representations, the model predicts the next word in a sequence.

The idea of using pre-trained embeddings came from Collobert and Weston in 2008 [Collobert and Weston, 2008]. The embeddings that they created display one of the most important characteristics of word embeddings, which is grouping together similar words in the vector space. They created the embeddings by training their language model on a large database and used them to initialize the lookup table of the multi-task training procedure. The procedure aims to learn multiple tasks at the same time, by following a few simple steps in a loop: a task is selected together with a sample of its training data and the neural network weights are updated based on this task. The lookup table (i.e. embedding matrix) is shared among the tasks.

Research in the area of word embeddings was significantly pushed forward by Mikolov et al., when they introduced **word2vec** [Mikolov et al., 2013b]. Similarly to Collobert and Weston, the team started with a neural language model which they adjusted to be more efficient and generate higher quality embedding vectors. In the course of two papers, the authors propose two architectures for learning word embeddings and methods to improve training speed and accuracy. The architectures have different context representations and different learning objectives. They are based on language models which randomly initialize all the word vectors, with **continuous-bag-of-words (CBOW)** predicting the word using its context, while **skip-gram** does the opposite, predicting the context based on the word. As the models do not contain any non-linear hidden layer, they are computationally less expensive. The main goal behind the two architectures was "to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words" [Mikolov et al., 2013a].

When analyzing the results of their architectures, the authors observe that their embeddings are able to capture complex semantic relationships. With simple algebraic operations (such as addition), the vector of one word can be obtained based on its relationship with other words. The example that the authors give is that of the relationship between the pairs of words 'big'-'biggest' and 'small'-'smallest'. The word vector for 'smallest' can be obtained in the following way:

$$X = \text{vector('biggest')} - \text{vector('big')} + \text{vector('small')}$$

The word vector closest to X based on cosine distance, will be the vector for 'smallest'. With more data, even more sophisticated relationships such as country-capital can be captured (e.g. Germany-Berlin, Romania-Bucharest). The embeddings have a high dimensional representation (between 50 and 300 dimensions), which when reduced to two dimensional space (for example through Principal Component Analysis), would look like in Figure 2.2.

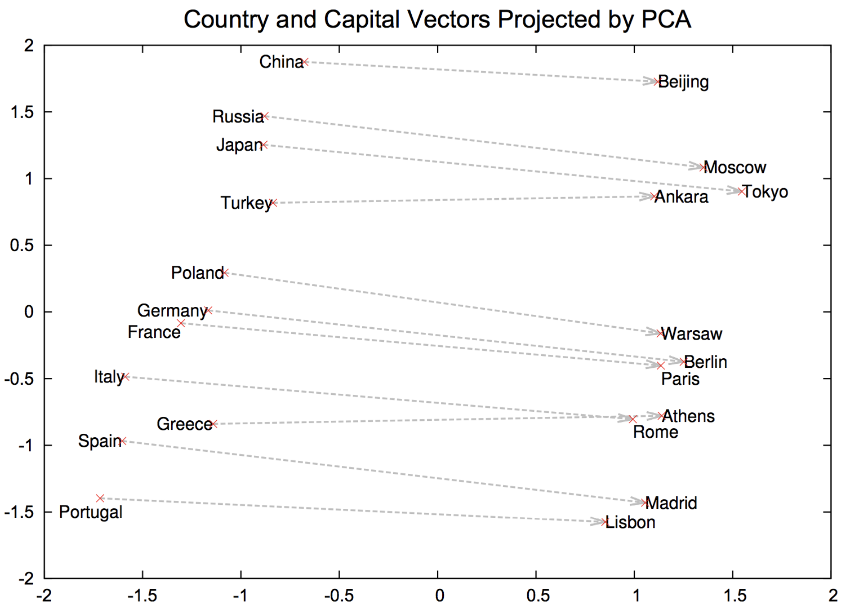


Figure 2.2: Relationship between country and capital cities two dimensional space [Mikolov et al., 2013a].

Word2Vec is an important milestone for NLP research and has created the context for designing new and improved word embeddings. In the course of this thesis, we experimented with newer pre-trained embeddings, including GloVe [Pennington et al., 2014], and FastText [Bojanowski et al., 2016], as well as the Universal Sentence Encoder (USE) [Cer et al., 2018] for sentence embeddings. We describe these embeddings in Sections 2.4.3 to 2.4.5

2.4.3 GLOVE embeddings

In this section we introduce GloVe [Pennington et al., 2014] embeddings which we use in our experimental framework as text representation for configuring the baseline. The use of these embeddings is also wide-spread in the deep learning literature for duplicate question detection, most researchers reporting their results with GloVe ([Wang et al., 2017], [Gong et al., 2017] [Kim et al., 2018]).

In 2014 Pennington et al. introduced GloVe: Global Vectors for Word Representation. The authors aimed both to capture meaning in vector space, as well as leverage the power of global statistics, instead of just the local context. Word2Vec takes into consideration only local context (the window of n words surrounding the target word), failing to recognize if two words occur together simply because one of them is very common (like 'the') or there is connection between the words. Thus, GloVe combines the two main approaches for learning word vectors: 1) global matrix factorization methods, such as Latent Semantic Analysis (LSA)[Deerwester et al., 1990] and 2) local context window methods, such as Skip-Gram [Mikolov et al., 2013a]. LSA learns significant statistical information, but is not able to capture the underlying vector-space structure. Skip-Gram does the opposite, missing out on the potential of global statistics.

GloVe builds a co-occurrence matrix using a fixed windows size. This leads to local context being taken into account. The co-occurrence of two words is the number of times they appear in the same window. The authors prove that the *ratio* of the co-occurrence probabilities of the two words (instead of simply the co-occurrence probabilities) contains valuable information and aim to encode this aspects in their vectors. The model is trained on global co-occurrence counts of words and minimizes a weighted least-squares error, producing a word vector space with meaningful substructure.

The authors also show that GloVe produces better embeddings, faster than Word2Vec. GloVe and Word2Vec have since been proved to have roughly the same performance on downstream tasks. In our work, we evaluate how GloVe text representation compares to newer approaches to word embeddings. In addition, we use GloVe in evaluating the baseline and in the experiments where we assess the performance of different components of our solution.

2.4.4 Fast Text embeddings

FastText is an efficient NLP library introduced by Facebook AI researchers in 2016 [Bojanowski et al., 2016], an extension of the Word2Vec research. Their main goal was to improve upon the continuous skip-gram model [Mikolov et al., 2013a] by taking into consideration subword information. Simply put, their new approach involves learning word representations as the sum of character n-grams learned vectors. We include FastText word embeddings as possible text representation in our experimental framework, as subword information is supposed to alleviate the problem of noisy data and help with rare words.

In order to capture internal word information, words are represented as bag-of-character n-grams, where n ranges from 3 to 6. The reason behind the size of the n-grams lies behind the fact that these lengths can capture valuable information, ranging from suffixes to longer roots. As an example, assume we have the word "where" and $n = 3$. Thus the word can be represented as: $< wh, whe, her, ere, re, where >$. As we can notice the full sequence is also included, which means that a representation for the whole word is also learned. For each n-gram found from the training corpus a vector representation is learned. Finally, the word is represented as the sum of vector representations of its n-grams.

FastText provides the following advantages over word2vec:

- it is able to deal well with out-of-vocabulary words (words that do not appear in the training corpus). These words are split into n-grams and their vector representation is averaged. This method works, because even if a word as whole may not be present in the vocabulary of the embeddings, it is quite possible that the n-grams that compose it have already been encountered.
- it performs well on creating better vector representations for words that occur rarely in a training corpus (such as disease names, technical terms etc.). The reason is same as above: the word can be split into n-grams and their representation is averaged.

Thus, FastText has the advantage of creating a meaningful representation for rare and out-of-vocabulary words, instead of initializing their vectors with zeros or random numbers. We want to leverage this information in our experiments, especially since we have not found many papers reporting the use of FastText pre-trained embeddings. Considering that in Q&A users might make spelling mistakes (making data quite noisy) it will be interesting to assess how these embeddings perform.

2.4.5 Sentence Embeddings

Word embeddings have been the norm for many deep learning NLP tasks, paving the way to the idea of Universal Embeddings (pre-trained embeddings that can be used in a wide range of downstream tasks). These methods perform a shallow form of transfer learning, while recent research in the area of embedding sentences has shown promising results and stronger transfer ability between different tasks [Conneau et al., 2017].

Pre-trained sentence embeddings follow the line of work on word embeddings, as previous success on word-level data motivated the idea of generating embeddings for longer pieces of text (such as sentences or paragraphs). We will briefly describe some of the initial approaches, before discussing the Universal Sentence Encoder [Cer et al., 2018] created by Google.

The trivial way to obtain sentence representations is by averaging the word vectors of the sentences. However, this approach had an overall poor performance demonstrating the need for more sophisticated techniques. Thus, the traditional average was replaced by a weighted average and its modification by principal component analysis (PCA) [Arora et al., 2016]. Authors claimed a 10% to 30% improvement in performance for sentence similarity tasks, leading to the establishment of a strong baseline. The weighting procedure is called *smooth inverse frequency.* A further improvement in performance is brought by the work of Rücklé et. al [Rücklé et al., 2018]: they propose the concept of *p-mean word embeddings* and concatenate different types of such vectors which contain syntactic, semantic or sentiment information. P-mean stands for *power mean* which is a natural generalization of the arithmetic mean [Hardy et al., 1988]. This method outperforms the previous baseline.

Cer et al. introduce the Universal Sentence Encoder [Cer et al., 2018] in early 2018. The team from Google creates two architectures aiming at generating a fix-sized sentence encoding, each with a different goal in mind: high accuracy at the expense of increased complexity or high efficiency at the cost of (a slightly decreased) accuracy. The two models are:

- **Transformer Encoder**. This sentence encoding model leverages the encoder from Transformer architecture introduced by Vaswani et at. [Vaswani et al., 2017]. The key component of the Transformer is the *Multi-attention head*, which helps the model "attend" to the relevant information. Basically, the model computes multiple attention weighted sums, learning "context aware representations" [Cer et al., 2018].
- **Deep Averaging Network (DAN)**. This technique computes the average of the input embeddings of words and bi-grams and passes the results through a feedforward deep neural network. The output of the DNN is the sentence embeddings. This approach is simpler and requires less complexity and resource consumption.

The output of both networks is a 512 dimensional vector, representing the sentence embedding. The Universal Sentence Encoder is trained jointly in an unsupervised manner with a large corpus of text and in a supervised manner on the SNLI [Bowman et al., 2015] dataset (similar to InferSent [Conneau et al., 2017]). The experiments presented in the paper prove that sentence embeddings outperform the word embeddings, although no comparison between other baselines or previous work is discussed. The DAN pretrained model that generates sentence embeddings is published so that it can be used in further research.

We are interested in the impact that the Universal Sentence Encoder can provide in the context of duplicate question detection. The deep learning networks that we have chosen for our framework model the relationship between the word embeddings belonging to the question pair, which is why we incorporate the embeddings generated by the Universal Sentence Encoder as an additional feature before the classification layer.

2.4.6 Evaluation of word and sentence embeddings

In this section we discuss our choice of word and sentence embeddings and present a brief comparison between them.

In terms of **word embeddings** our choice consists of GloVe and FastText. GloVe was picked because it is the embedding of choice in all the models that were included in our experimental framework, thus enabling a direct comparison of our approaches with the original papers. FastText was included mainly for its ability to deal with out-of-vocabulary (OOV) and rare words, especially because our datasets are collected from real-life Q&A forums and misspellings are common. In terms of **sentence embeddings**, we have opted for the Universal Sentence Encoder (USE) due to the ease of integrating it within our framework. We used the pre-trained embeddings generated by DAN architecture. Moreover, what makes USE interesting to us is that it was shown to perform well on semantic relatedness and textual similarity tasks [Perone et al., 2018].

Perone et. al present an empirical evaluation of the most important advances in sentence embeddings [Perone et al., 2018]. Among state-of-the-art techniques such as p-mean, USE they also include word embeddings (Word2Vec, GloVe, FastText) and average them in order to obtain sentence embeddings. Inspired by this paper, we present Table 2.1, which shows an overview of the embeddings that we choose for our work.

Name	Training Methods	Embedding Size
FastText (BoW, Common Crawl)	Unsupervised	300
GloVe (BoW, Common Crawl)	Unsupervised	300
USE (Transformer)	Unsupervised + Supervised	512

Table 2.1: High-level comparison of embeddings. The table summarizes the training methods used and the size of the resulting pre-trained embeddings.

2.5 Summary

In this chapter we have introduced necessary theoretical background information, describing the most important concepts related to Deep Learning in the context of Natural Language Processing. We have briefly discussed how a basic neural network works and what training in the context of DL means. Special emphasis was put on architectures that are relevant to this work and on the word embeddings chosen for the experiments. Specifically, our deep learning models include LSTM layers, GRU layers, attention mechanism, bidirectional and siamese networks. Furthermore, as discussed in Section 2.4.6, we use GloVe, FastText and Universal Sentence Encoder for text representation.

We aim to solve the problem of duplicate question detection with Deep Learning. The main advantage of this technology is its ability to learn from unstructured data and extract abstract features from raw text. In contrast, it can be sensitive to lack of data and we propose solutions to this problem in Chapter 4.

3 Related Work

The development of deep learning techniques for duplicate question detection has been an intensive area of research in recent years, especially since the public release of Quora Duplicate Questions Dataset [1] in 2017 and launch of the equivalent Kaggle competition [2], which challenges its users to solve this natural language processing problem. There have been many encouraging results, which we will overview in the following sections.

The main purpose of this chapter is to discuss related work. First, we report on state-of-the-art solutions for DQD; second, we introduce three similar NLP problems and discuss how they are related to the task at hand; finally, we describe the problem of duplicate detection in general. In the last section of this chapter, we discuss how this research is positioned in the literature. We also briefly present techniques used in the literature to improve the performance of different models and experiments that pushed the study of DQD forward.

3.1 State-of-the-art in DQD

In this section, we detail several state-of-the-art approaches and discuss how they compare to each other. The focus is on deep learning models, which can be classified into two main categories based on how the pairs of sentences are handled: **sentence encoding** and **sentence interaction**.

3.1.1 Sentence Encoding

Yu et al. define sentence encoding as a framework in which the model learns a pair of fixed-size representations for the input sentences and generally uses a comparison function to combine the vectors into a unified representation ([Yu et al., 2018]). The two sentences are encoded independently of each other. The main advantage of this technique is that parameters are shared across the network, consequently making the model easier to train. There are a number of works that can be classified into this category and their approaches consist of Siamese neural networks (see Section 2.3.5) which use convolution neural networks (CNN), LSTM, GRU or RNN cells to encode both sentences.

Bogdanova et al. are one of the first to tackle specifically the problem of DQD and propose a sentence encoding neural solution [Bogdanova et al., 2015]. Their **Deep Convolutional Neural Network (DCNN)** transforms words into embeddings and

[1]Details about dataset release: https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[2]Competition details: https://www.kaggle.com/c/quora-question-pairs

constructs distributed vector representations for questions, which are scored using a similarity metric. They also explore the impact of domain-specific word embeddings and training set size. Their results show that CNN performs better than traditional methods based on Jaccard coefficient and SVM, especially on pre-trained in-domain data. They experiment with datasets from AskUbuntu and MetaExchange forums. The authors claim a 92% accuracy, fact which is verified by a reproducibility study [Silva et al., 2018]. The latter found that in the original work of Bogdanova et al., there was an error in the preparation of the data: specifically, they noticed that the questions contained a clue in the form of the string "Possible duplicate" plus a link to the question that is a duplicate. As the model is supposed to learn what duplicate questions look like, the authors claim that this error was at the basis of the 20% jump in accuracy, obtaining instead 73.4% with the same network. Furthermore, Silva et al. propose a hybrid DCNN which achieves 79% accuracy on AskUbuntu dataset.

Homma et al. developed **Siamese GRU)** and **Siamese RNN** (see Sections 2.3.3 2.3.1) architectures and experimented with different distance metrics on Quora dataset [Homma et al., 2017]. They showed that, based on only 10-15 words of a sentence, a GRU neural network is able to extract the majority of semantic meaning and created a technique for data augmentation which achieved promising results in reducing overfitting. Their best result was an accuracy of 84.95% with a network with two hidden layers and GRU cells for encoding the sentence pairs on the augmented dataset. The framework described in Chapter 5 implements this architecture and aims at reproducing this paper's best result. Implementation details are described in the corresponding Section 5.2.2.

3.1.2 Sentence Interaction

Sentence interaction [Yu et al., 2018] (also loosely referred as 'compare-aggregate framework' [Wang and Jiang, 2016]) aims at capturing the interaction between the input sentences. In general, these methods compare vector representations on a more granular level (for example words and phrases) and aggregate these results to output the prediction. The motivation behind this kind of approach is that a single, fixed-size vector, used for encoding a sentence, is not able to fully learn the complex patterns of a text sequence and important information may be lost.

The **Decomposable Attention (DecAtt)** model[Parikh et al., 2016] achieved state-of-the-art results on Natural Language Inference (NLI) problem (SNLI dataset, [Bowman et al., 2015]) when it was introduced. It is one of the first models to use attention mechanism (see Section 2.3.6) for soft-aligning the elements of the two sentences. The soft-alignment matrix is used to obtain the aligned subphrases, which will be compared. The vectors obtained from the comparison phase are aggregated through summation and used to obtain the prediction. The advantages of this approach is that it uses an order of magnitude fewer parameters than the other models it is compared to and it does not need to take into account word order, rendering this model very efficient. We have incorporated this model in our solution framework, mainly because of its simplicity and effectiveness, but also because this model is actively used up to date. Engineers from Quora experimented with this model for detecting duplicate questions posted on the website and obtained an accuracy of 86% with word embeddings pre-trained on Quora's text corpus. Furthermore, the winning team in the *Kaggle Quora Competition* included this model in their approaches. Thus, we find it relevant to our work. Implementation details are provided in Section 5.2.2.

The **Enhanced Sequential Inference Model(ESIM)** [Chen et al., 2016] achieved state-of-the-art results soon after DecAtt on the NLI problem. It surpassed the accuracy of DecAtt (86.3%) reaching 88.6% on SNLI dataset. This model is related to DecAtt, bringing some improvements. It uses a BiLSTM (see Sections 2.3.4, 2.3.2) to encode the input sentences, an attention layer, a local inference layer with subtraction and element-wise product of aligned sub-phrases, a BiLSTM inference composition layer, a pooling operation and, finally, an output layer. The aim of the authors was to enhance sequential inference models and capture richer information. This model is often used as baseline in other works ([Peters et al., 2018]) and it was also employed by the winners of *Kaggle Quora Competition*, along with DecAtt. Because of these reasons, we incorporate it in our framework. Further implementation details are provided in Section 5.2.2.

Wang et. al. achieved state-of-the-art results on DQD problem, with an accuracy of 88% on Quora dataset, from their **Bilateral Multi-Perspective Matching model (BiMPM)** [Wang et al., 2017]. Their model encodes and matches the two sentences bidirectionally (P against Q and Q against P) from multiple perspectives. The authors devise four methods for matching each time-step of P against all time-steps of Q and the other way around. The matching results are then aggregated with a BiLSTM in a fixed-length vector and the prediction is made based on this vector. The authors compare this technique to their previous work in the field and show that it outperforms Siamese CNN and Siamese LSTM by a large margin. The approach is tested on three different NLP problems, including Natural Language Inference, Answer Sentence Selection (what we refer to as Question Answering or QA in Section 3.2) and Paraphrase Identification (PI). This work is relevant to us because the authors release a training, development, test data split, which is used in most papers that came out after this one. We also adopt this data split in our experiments, thus having a good frame of reference for comparing results.

Another model, called **Densely Interactive Inference Network (DIIN)** produces improved results on NLI and paraphrase identification problem [Gong et al., 2017]. They introduce the concept of *interaction tensor*, which is an extension of multi-head attention [Vaswani et al., 2017]. This interaction tensor is able to capture rich semantic information by "building word-by-word dimension-wise alignment". For the paraphrase identification problem, the authors report an accuracy of 89.06% for simple DIIN and 88.9% for ensemble DIIN.

More recently, the **Densely-connected Recurrent and Co-attentive neural Network (DRCN)** model has achieved a new state-of-the-art performance on five different datasets [Kim et al., 2018], in the field of NLI, PI and QA. Their proposed architecture leverages the power of deep RNNs, which are better at learning information from longer sequences. The attention mechanism is used to retain co-attentive feature information through the whole network (from the first layer to the last one). The paper reports an accuracy of 90.15% for the simple DRCN model and 91.30% for ensemble DRCN on Quora dataset.

3.2 Natural Language Sentence Matching

Natural Language Sentence Matching (NLSM) [Wang et al., 2017] is the problem of identifying the relationship between two (or more) input sentences. Thus, modeling a pair of sentences is fundamental to many NLP tasks, with application to real-world use cases (such as online forums, community question answering websites and chatbots). Paraphrase Identification (PI), Natural Language Inference (NLI), Question Answering (QA) are some of the problems that fit in the category of NLSM. In terms of deep learning approaches, the trend has been to design architectures that are able to generalize well and tackle more than just one sentence pair modeling task, as we see in Section 3.1.

Paraphrase Identification is the task of checking whether two sentences are paraphrases i.e. they express the same meaning. Paraphrases can be seen as the equivalent of lexical synonyms, but at sentence level, which means that both sentences must be true under the same conditions. The release of **Microsoft Research Paraphrase Corpus (MSRP)** [Dolan and Brockett, 2005] has been a major step forward in this area as it enabled researchers to train and test their models on a standard corpora. Duplicate Question Detection is a subproblem of this category as it entails identifying whether two questions are looking for the same answer [Bogdanova et al., 2015]. Duplicate questions are harder to identify than duplicate declarative statements [Rodrigues et al., 2018] . Table 3.1 shows some examples of paraphrase and not paraphrase sentences from MSRP dataset, with varying degree of word overlap.

Sentence 1	Sentence 2	Label
Charles O. Prince, 53, was named as Mr. Weill's successor.	Mr. Weill's longtime confidant, Charles O. Prince, 53, was named as his successor.	Paraphrase
The euro rose above US1.18, the highest price since its January 1999 launch.	The euro rose above 1.18the highest level since its launch in January 1999.	Paraphrase
The Gerontology Research Group said Slough was born on DATE, making her NUMBER years old at the time of her death.	"Mrs. Slough" is the oldest living American as of the time she died, L. Stephen Coles, Executive Director of the Gerontology Research Group, said DATE.	Not Paraphrase

Table 3.1: Examples of MSRP dataset [Dolan and Brockett, 2005].

There has been extensive work on the problem of **Natural Language Inference (NLI)**, especially since the release of Stanford Natural Language Inference (SNLI) Corpus [Bowman et al., 2015]. The corpus contains $570k$ pairs of English sentences, which were manually labeled as *entailment, contradiction or neutral.* Correctly identifying the label is challenging as it requires a very good understanding of the context and the ability to logically reason between a premise sentence and a hypothesis sentence. To clarify, the main task in this case is to figure out whether the hypothesis can be inferred from the premise. NLI is closely related to PI, mainly because paraphrases can be seen as a special case of symmetrical entailment. In the case of entailment, one sentence is true in the context of the other, but the reverse does not hold. For example, assume we have the following sentences from the work of Parikh et al.: ”Bob is in his room, but because of the thunder and lightning outside, he cannot sleep.” and ”Bob is awake.” Clearly, the second sentence follows the first one, but the opposite does not hold, as just by looking at the second sentence we do not know why Bob is awake. If the second sentence was ”Bob is awake because of the storm” then the relationship between the two sentences would be symmetrical and they would be paraphrases. Table 3.2 shows some examples of Premise and Hypothesis sentences and their label.

Premise	Hypothesis	Label
A soccer game with multiple males playing.	Some men are playing a sport.	Entailment
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping	Contradiction
An older and younger man smiling.	Two men are smiling and laughing at the cats playing on the floor	Neutral
A black race car starts up in front of a crowd of people.	A man is driving down a lonely road.	Contradiction

Table 3.2: Examples from SNLI dataset [Bowman et al., 2015].

Question Answering and Information Retrieval (QA) or Answer Sentence Selection [Yang et al., 2015] is defined as the task of ranking candidate answers for a question based on how well they answer the question. The relevance between question-answer pairs can be determined by their similarity [Wang et al., 2016], similarly to PI. One challenge that arises in a QA setting is that there may be very little word overlap between query-answer pairs, what is also known as *lexical chasm* [Berger et al., 2000]. Efficiently solving this NLP problem is important for maintaining the quality of Q&A websites as thousands of questions and thousands of (more or less vague) answers are posted everyday. In a real world scenario this is a two-step task: first identifying whether this question has been asked (i.e. detecting duplicate questions), second, ranking the available answers based on their correctness and relevance. Table 3.3 shows an example of a question and a set of candidate answers. On some Q&A websites users have the possibility to vote for the best answer, but this might not always be the most appropriate one as users might choose to vote a funny/unrelated answer to a question.

In general, all these problems (PI, NLI, QA) can be solved in a similar way to DQD, by converting the text data in a numerical, vector-based representation and afterwards ap-

Question	Sentence	Label
Q1. How are glacier caves formed?	D1-0. A partly submerged glacier cave on Perito Moreno Glacier.	0
	D1-1. The ice facade is approximately 60 m high	0
	D1-2. Ice formations in the Titlis glacier cave.	0
	D1-3. A glacier cave is a cave formed within the ice of a glacier.	1

Table 3.3: Example from WikiQA dataset [Yang et al., 2015]. Answer D1-3 has label 1 which means this is the correct answer.

plying different operations and transformation strategies for adding or removing features for identifying similarity and semantic relationship, before finally outputting a prediction. Models such as BiMPM [Wang et al., 2017], DIIN [Gong et al., 2017], DRCN [Kim et al., 2018] address multiple of these tasks at the same time, with DRCN achieving the overall state-of-the-art.

3.3 Duplicate Detection

The problem of duplicate entries that refer to the same real world entity has existed for decades, especially in databases. Different names are attributed to it, from *record matching, name matching* to *duplicate detection*. Elmagarmid et al. refer to it as **duplicate record detection** and define it as "the process of identifying different or multiple records that refer to one unique real-world entity or object" [Elmagarmid et al., 2007] .

The first step in detecting duplicate records in a database is to prepare the data, so that comparison is easier to pursue. The steps to achieve more usable data include parsing, transformation (data type conversion) and standardization (formatting fields in a standard way).

Once the records are easier to use and manipulate, techniques for measuring similarity can be applied. According to Elmagarmid et al., the methods can be categorized in the following way:

- **Field matching**. These techniques are based on string comparison and aim to fix different kinds of errors. In this category we have *character-based similarity metrics [Jaro, 1980], token-based similarity metrics [Gravano et al., 2003], phonetic similarity metrics [Philips, 2000].*

- **Probabilistic matching models**. These techniques encompass probabilistic and supervised machine learning methods. Approaches include different Bayesian models ([Winkler, 2002], [Verykios and Moustakides, 2004]).

- **Supervised and semi-supervised learning**. Algorithms such as *CART and SVM* perform well on matching duplicate records from multiple fields ([Cochinwala et al., 2001], [Bilenko et al., 2003]).
- **Active-Learning-Based Techniques** These techniques are used to identify pairs of duplicates which are more challenging to correctly classify, but would help the algorithm generalize better [Sarawagi and Bhamidipaty, 2002].
- **Unsupervised Learning**. When there is not enough good training data available, methods such as clustering perform well on detecting duplicate records ([Ravikumar and Cohen, 2004]).

More recent techniques and toolkits include DuDe, Magellan, and JedAI. DuDe [Draisbach and Naumann, 2010] is an extensible toolkit for duplicate detection, which supports multiple data sources (e.g. relational databases, CSV files) and similarity measures. Magellan [Konda et al., 2016] is an entity matching system that covers the whole record linkage pipeline (blocking, matching, exploring, cleaning, debugging, sampling, labeling, estimating accuracy) and can be easily integrated with other Python packages. JedAI [Papadakis et al., 2018] is a fast deduplication tool that can handle both structured and semi-structured data.

Overall, duplicate record detection is related to duplicate question detection on a high level, but the techniques are not suitable. One of the main reasons behind this is the complexity of natural language compared to structured data which can be found in a database, thus requiring more sophisticated approaches.

3.4 Positioning of our work

The problem of duplicate question detection has been consistently studied in the recent years, with solutions relying on feature engineering and classic machine learning algorithms ([Kozareva and Montoyo, 2006]) or advanced deep learning architectures. An overview of the models discussed in Section 3.1.1 and Section 3.1.2 is presented in Table 3.4, similarly to the comparison done by Lan and Xu [Lan and Xu, 2018]. We have also included our best performing solution in this table. All these models, except **SiameseGRU**, used the same train/dev/test split of Quora dataset, provided by the creators of BIMPM model [Wang et al., 2017], which makes their results comparable. **DecAtt, ESIM and SiameseGRU** have been included in our experimental framework, so that both sentence encoding and sentence interaction approaches are covered, but also because these models showed promising results and as still relevant today. We will discuss the implementation of the framework with state-of-the-art deep learning models in Section 5.2.2.

As discussed in the previous sections, the trend in solving the DQD problem (and other similar NLP problems) has been to devise improved and more sophisticated architectures to tackle the challenge of semantic equivalence. For this purpose different strategies for encoding and representing sentences have been created, different ideas for capturing

Approach	Sentence Encoder	Interaction & Attention	Aggregation & Classification
DRCN [Kim et al., 2018]	BiLSTM	RNN + soft alignment + concatenation + autoencoder	maxpooling + MLP
DIIN [Gong et al., 2017]	highway network + self-attention	dot product + DenseNet [Huang et al., 2017]	maxpooling + MLP
BiMPM [Wang et al., 2017]	BiLSTM	multi-perspective matching	BiLSTM + MLP
DecAtt [Parikh et al., 2016]	-	soft alignment + dot product	summation + MLP
ESIM [Chen et al., 2016]	BiLSTM before and after attention	soft alignment + dot product + subtraction	average and max pooling + MLP
SiameseGRU [Homma et al., 2017]	GRU	-	dot product + squared difference + MLP
SiameseLSTM [Wang et al., 2017]	LSTM	-	cosine similarity + MLP
SiameseCNN [Wang et al., 2017]	CNN	-	cosine similarity + MLP
DecAtt + ESIM + USE sentence embeddings + data augmentation (our solution)			

Table 3.4: Summary and comparison of models based on accuracy. MLP stands for Multi-layer Perceptron (see Section 2.2.1 for reference)

interaction have been tested and different variants of attention have been implemented as discussed in Section 3.1.

Other researchers have conducted experiments to understand different aspects of DQD, by using already existing architectures (DCNN [Bogdanova et al., 2015]). For example, Saedi et al. studied the impact of the training set size on the performance of different models, in the context of duplicate question [Saedi et al., 2017]. They concluded that traditional methods, such as Jaccard coefficient, perform better on smaller training sets, while more complex deep learning techniques (such as DCNN [Bogdanova et al., 2015]) are superior on sufficiently large datasets. The research of Rodrigues et al. is also interesting, as they explore the impact of training sets which consist of generic-domain, well-edited questions versus narrow-domain, poorly formulated questions [Rodrigues et al., 2017]. They also focus on the problem of transfer learning, concluding that there is a negative impact on accuracy when the algorithms are trained on data from multiple domains and then applied to a narrow domain.

The solution presented in this work, in Chapter 4, studies different approaches for text representation, data augmentation, transfer learning and ensembling that can be applied to improve the performance of deep learning architectures in the area of DQD. For this reason, the experimental framework includes three well-performing models and different strategies to enhance their performance. We study how smaller samples of competitive DQD datasets with different characteristics behave with these models and new components.

4 Solution

In this chapter, we describe the limitations encountered in the study of DQD and our proposed approaches to solve these issues. The main problem that we are aiming to solve is that of limited *labeled* training data typically available for research or production use in the area of DQD. In this case, manually labeling data requires tremendous human effort for reading and annotating pairs of questions (normally by more than one person) in order to obtain a high quality dataset. The alternative would be a sophisticated sampling technique/machine learning model that automatically labels data, including training data. Unfortunately, this approach carries the risk of creating a noisy dataset, which would require manual validation. Either way, the process is expensive and time-consuming. Thus, the challenge becomes either to find suitable methods for enhancing available data or to identify which techniques empower classifiers to efficiently learn from less data. First, we motivate the need for techniques for smaller datasets and the problems that need to be addressed in this setting. Next, we introduce the solutions that we undertake in this thesis. We propose to solve the problem of reduced amounts of training data through finding suitable text representation for questions, data augmentation, transfer learning and ensembling. With these approaches, we aim to build a model that requires less training data for achieving good performance.

4.1 Foundations

The main limitation of the models and techniques proposed so far and described in Chapter 3 is that they assume an unrealistic scenario, where labeled training data is abundant. Current solutions have mainly competed against each other in increasing accuracy on big datasets by designing sophisticated architectures that are data hungry and often result in overfitting smaller datasets.

Unfortunately real-life applications deal with noisy and limited datasets, sometimes targeted to a specific domain and with a distinct vocabulary. The effect of using different DL models on datasets with this kind of characteristics has not been thoroughly studied. Hence, the users who dispose of a small training sample and want to create a system that can detect duplicate questions will find themselves at a loss in the myriad of tools and techniques available, none of them tailored for their use case. Thus the goal of our research becomes:

- Study techniques that enable DL models to efficiently learn from limited training sets by enhancing available data and models.

In order to solve the main problem of limited data for DQD and answer the research questions formulated in Chapter 1 we formulate four different subproblems and their inherent research questions.

The first problem that we aim to solve is concerned with determining which text representation is the most robust to limited training sets. We propose evaluating state-of-the-art word embeddings and sentence embeddings and identifying which are most suitable for detecting duplicate questions. This leads to our first research question:

1. Which text representation is the most beneficial for identifying duplicate questions from a small sample size?

The second problem refers to the tendency of DL models to overfit small datasets which poses the challenge of how to generate more training data. We propose several data augmentation techniques that can create new data and help the models generalize better. The second research question that we address is:

2. How can we generate new training data such that the DL models are able to generalize better?

The third problem addresses the situation where a bigger, related dataset is available and our target dataset is small. We introduce two ways to transfer information between the two datasets, so that the learning process does not begin from scratch. This approach covers the scenario in which we have a big dataset from a different domain available and less data from our target domain. The third research question can be formulated as follows:

3. How can transfer learning help when there is not enough training data available?

Finally, the last problem refers to the strengths and weaknesses of our DL models and the different perspectives that they tackle (sentence encoding and sentence interaction). We propose to ensemble them by stacking, such that the different knowledge that they gained through training can be leveraged. Thus, we aim to answer the final question:

4. How can ensembling help in better detecting duplicate questions using a small dataset?

Our proposed solution uses a default setting which includes three state-of-the-art deep learning models. We chose not to design a new DL architecture, but instead to find methods that can be used on top of these models, in order to consistently improve their performance. We believe such an empirical study is necessary because there is a wide range of techniques available which have been assessed on increasingly big datasets, overseeing the fact that in some fields text data is still limited. What is more, there is no unified framework that evaluates these techniques and can determine whether they are dataset agnostic or dataset specific.

4.2 Text Representation

To the best of our knowledge the existing solutions in the area of DQD lack a detailed analysis of how using different pre-trained word and sentence embeddings affect the performance of the classifiers. This makes it difficult to have a general understanding of the real impact the available pre-trained embeddings have on the final classification result. The goal of this section is to outline the problems related to text representation and the proposed methods consisting of two pre-trained word embeddings, combined with sentence embeddings.

The works described in Chapter 3 mainly use Glove [Pennington et al., 2014] word embeddings for initializing the embedding layer in their architectures. As explained in Section 2.4.3, GloVe builds word vectors based on the idea that if words appear in the same contexts they tend to have the same meaning. GloVe has limitations when it comes to how it deals with rare or out-of-vocabulary words, providing no embedding for them. As datasets tend to be noisy, especially in online user forums where misspellings occur and users might coin new terms or ask very technical questions, this is a disadvantage. For example, in the question "How do i get nvidia gt650m working?" taken from AskUbuntu dataset (see Section 5.1.2), GloVe would only find word embeddings for "how", "do", "i", "get", "working", which makes the initial sentence lose some its important details.

There are novel approaches that address the shortcomings of GloVe. FastText [Bojanowski et al., 2016] is trained in an unsupervised manner, and learns representations for n-gram units and then sums them to get the word vector. Hence, it has the advantage of better dealing with rare and out-of-vocabulary words (see Section 2.4.4). On the other hand, the recently released Universal Sentence Encoder (USE) [Cer et al., 2018] is trained both in a supervised and unsupervised manner and builds fixed-size sentence embeddings(Section 2.4.5), claiming to outperform word embeddings. We summarize these new techniques in Table 4.1.

There has not been much research into how these newer technologies can benefit the study of DQD. We propose a comparative analysis between them, treating the sentence embeddings as a feature to be concatenated to the word representation. The reason behind this is that the architectures that we chose to work with in our experimental framework focus on the words, which are lost when we obtain the sentence representation.

We experiment with both GloVe and FastText to build the embedding matrix and then concatenating the USE sentence embeddings to the word representations. We will have four possible scenarios for text representation: GloVe, FastText, GloVe and USE, FastText and USE. Typically two sentences that are similar in terms of semantics will have similar encoding and thus sentence embeddings should act as a strong feature in classification. We show that an additional feature from the USE brings a small boost in the performance of deep learning models. Based on this experiment we can answer the question *Which text representation is the most beneficial for identifying duplicate questions from a small sample size?* by claiming that GloVe and FastText are competitive with each other, while an additional feature from USE is beneficial.

Pre-trained embedding	**Description**	**Dimension**
GloVe [Pennington et al., 2014]	It contains 2.2M word vectors trained on Common Crawl.	300
FastText [Grave et al., 2018]	It contains 2M word vectors trained with subword infomation on Wikipedia 2017 and Common Crawl.	300
Universal Sentence Encoder [Cer et al., 2018]	Trained in an unsupervised way with data from Wikipedia, web news, web question-answer pages and discussion forums. Additionally, trained in a supervised way with SNLI corpus.	512

Table 4.1: Description of pre-trained word and sentence embeddings used in this research. FastText also takes into account the morphology of words by adding sub-word information to build the embeddings. Universal Sentence Encoder is a model that transfers the knowledge learned from big corpora of data to encoding the sentences it is fed.

4.3 Data Augmentation

Generating additional data in a meaningful way, to help classifiers generalize better, is challenging. This is a necessary step especially when only small datasets are available for training and the models are data hungry, tending to overfit the available data. In the context of image processing, typical data augmentation techniques include flipping and rotating images or removing some pixels, as essentially these methods do not change the original image. This is not possible in NLP, as adding or removing words or changing the order of words in a sentence might modify its whole meaning and end up introducing noise. In the context of DQD, we propose three methods that create varying amounts of new data and one method to add perturbation to the word embedding matrix.

4.3.1 Reverse Questions Augmentation

The first method of creating new data is inspired by the work Homma et al. and it entails reversing the order of questions. With this method we basically double the size of the original dataset, while maintaining the initial distribution of labels. The authors claim that this method is promising for siamese networks with asymmetrical operations such as subtraction [Homma et al., 2017], so it is be interesting to see how sentence interaction models are influenced by this augmentation technique. Table 4.2 illustrates an example of this rather simple technique. We show that this technique improves the performance of our models on limited amounts of data, where the questions have high word overlap, by a small margin.

ID	Question 1	Question 2	Label
1	How do you properly tie ice skates?	How can I tie my ice skate shoelaces properly?	Duplicate
2	How can I tie my ice skate shoelaces properly?	How do you properly tie ice skates?	Duplicate

Table 4.2: Example of reversing question order data augmentation technique. The first pair is the original one and the second pair is added as augmentation by reversing the order of questions.

4.3.2 Thesaurus Augmentation

The second method involves using an external resource such as WordNet[1] for finding synonyms for words in a given sentence and replacing them. Wordnet [Miller, 1995] "is a large lexical database of English, where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept". Replacing words with their synonyms is also known under the name of *Thesaurus augmentation* [Zhang and LeCun, 2015]. If a word does not have synonyms we leave it unchanged. This technique doubles the size of the original dataset. It is promising because it adds variety to the original dataset and exposes the neural network to new words. Nonetheless, this method bears the risk of introducing unwanted noise in the dataset, as very rarely words have exact synonyms. Table 4.3 continues the example from above, this time in the context of replacing synonyms. While this method is promising, we show that it does not give consistent results.

ID	Question 1	Question 2	Label
1	How do you properly tie ice skates?	How can I tie my ice skate shoelaces properly?	Duplicate
2	How do you in good order tie ice skate?	How can I tie my ice skate shoelace decent?	Duplicate

Table 4.3: Example of Thesaurus-based data augmentation technique. The first pair is the original one and the second pair is added as augmentation by replacing word with their synonyms. As we can notice, the replacement can sometimes sound quite unnatural, and even perturb the original label.

4.3.3 Heuristic Augmentation

We propose a a third method that entails sampling the available dataset based on a defined heuristic for word overlap and adding that sample again, but in reverse order. The assumption behind this idea is that we can help the model put more focus on

[1]https://wordnet.princeton.edu/

certain types of questions, selected based on that heuristic. In this method we are mostly interested in ambiguous pairs of questions, that could hypothetically make the classifiers more robust to real-life scenarios. We define a scoring metric for measuring word overlap: the ratio between the number of unique words that the two questions have in common and the total unique words. For this metric we take the lemma of the word and ignore the stop words. We want our models to be able to correctly classify pairs of questions that have a big share of vocabulary in common, but in fact have different intents (so non-duplicates). Furthermore, models should be able to detect complex paraphrases, where there is little word overlap but the questions are actually asking for the same thing. We establish a threshold for word overlap in these two cases and add these pairs again, in reverse order. An example of such ambiguous pairs of questions is shown in Table 4.4. We show that, even though this technique is not suited for data augmentation, it provides a good way to test the lexical knowledge of our models, as questions with high overlap are generally challenging to be correctly classified.

ID	Question 1	Question 2	Label
1	What was the immediate cause of the First World War ?	What were the causes of the First World War ?	Non-Duplicate
2	Is it true goats will eat anything ? How do they do it ?	What is the reason why goats are able to eat everything ?	Duplicate

Table 4.4: Example of sentences chosen based on word overlap heuristic. The first pair has high word overlap, but is in fact non-duplicate, while the second pair has low word overlap and is duplicate.

4.3.4 Gaussian Embedding Perturbation

Finally, the fourth strategy addresses the continuous space of word embeddings, instead of the discrete space of words. Following the work of Zhang et al., we modify the word embedding layer by adding small amounts of random Gaussian noise to it [Zhang and Yang, 2018]. This method, also known as *word embedding perturbation* has shown potential for sentence classification, which makes it interesting to try in our use case. We transform the embedding layer X_{emb} with e sampled from the Gaussian distribution:

$$X_{emb} = X_{emb} \odot e, e \sim N(I, \sigma^2 I)$$

This experiment shows us that this type of augmentation makes sentence encoding models more robust to noise, while the results vary for sentence interaction models.

To summarize, our experiment with data augmentation techniques encapsulates three methods that augment the dataset in the discrete space of words and one method that deals with the continuous space of word embeddings. The results of this experiment help us answer the question *How can we generate new data such that the DL models are able to generalize better?* by stating that adding the questions in reverse is a promising technique and fine-tuning the amount of Gaussian noise added to the embedding matrix can make models more robust. Overall, these methods only marginally improve the baseline, which suggests that more data needs to be generated for better improvements.

4.4 Transfer Learning

Deep learning models typically require large amounts of data in order to generalize well on unseen samples. These high quality datasets are not always easily available, which is also the use case we are exploring in this thesis. On the other hand, when a bigger, similar dataset is available, this problem can be mitigated through a technique called transfer learning. We propose two different methods for applying this technique to DQD. The basic idea for employing transfer learning methods is to take advantage of the weights learned by a DL model and apply them to a different dataset. This means using a model trained to solve a problem, on a related problem, which is efficient computational- and time-wise. Overall, transfer learning has the potential to make classification easier.We perform transfer learning for two reasons:

1. It is an effective technique for identifying models that are able to generalize well on related problems, thus saving resources.

2. When data from a domain is scarce, the model can be trained on a similar dataset and then applied to the target dataset (also known as domain adaptation).

The usage of pre-trained word embeddings is a transfer learning technique itself, as it entails gaining knowledge of semantics, syntax and context from a large corpus of text, before applying them to a specific task. The embeddings can then be used in a diverse range of NLP problems, including DQD. On the other hand, this approach can be quite shallow as embeddings are only loaded in the first layer of a network, while nowadays networks typically consist of many layers. From this point on, it depends on the architecture to learn as much as possible from the embeddings assigned to it. In more recent works, researchers are focusing on transferring entire models between different tasks ([Howard and Ruder, 2018], [Radford et al., 2018]). The most important advance in this sense is the concept of pre-training an entire language model, which has been overwhelmingly efficient in transferring knowledge between NLP tasks [Howard and Ruder, 2018]. This is mainly because the pre-trained language model has a deep understanding of grammar, semantics and other NLP issues. The idea has been further improved by building a task-agnostic pre-trained language model [Radford et al., 2018].

The models in our experimental framework are specialized in modeling the relationship between pairs of sentences. Hence, they are not task agnostic because they require that the data they are fed is a pair of sentences. For example, these models cannot be applied to sentiment analysis task, where the inputs are typically paragraphs. It is interesting to explore the models' ability to transfer information between related tasks and datasets in this more specific setting.

We propose to apply transfer learning in two different ways: first method consists of fully training and tuning a model on a dataset and evaluating on a different dataset; second method corresponds to pre-training the layers of the model on a dataset and fine-tuning and evaluating it on a new dataset. Both methods will be tried in two different situations: 1) when dealing with the same task, but a different word distribution and domanin; 2) when dealing with a different, but somehow related task. Theoretically, the pre-training and fine-tuning method is expected to perform better, potentially from less data, as the model does not need to learn everything from scratch. We show that this is true for the sentence encoding model, which definitely benefits from large amounts of information learned in the previous task. On the other hand, entire model transfer is not successful in this context. This effectively answers the question *How can transfer learning help when there is not enough training data available?*

4.5 Ensembling

The architectures applied to the problem of DQD are typically tackling the pairs of questions from two different perspectives: sentence encoding and sentence interaction (see Section 3.1), each with its own downsides. We propose to reduce the number of errors made by the models by ensembling them. The most popular ways to ensemble models are:

- Bagging. Build several models of the same type using different subsamples of the same training set. Reduces variance.
- Boosting. Build several models of the same type, where each model focuses on the error made by the previous one. Reduces bias.
- Stacking. Build several models of different types and average their predictions for increasing their overall predicting power.

Sentence encoding models have a tendency to generalize better across different datasets, but because of their simplicity they sometimes lack the ability to understand contexts. On the other hand, sentence interaction models pay more attention to the relationship between words in a sentence and when the sentences have a high overlap, they are left with very few keywords to focus on [Lan and Xu, 2018]. Tables 4.5 and 4.6 show some examples of errors made only by sentence interaction, respectively only by sentence encoding models.

In order to improve the performance of our models, it would be better to leverage the knowledge each model gained separately by training and use it to make better predictions. Our approach consists of creating an ensemble that includes the three models we proposed in the experimental framework. We show that the ensembles perform better than any individual model. On the other hand, the best performing ensemble contains only the sentence interaction models, as the sentence encoding model underperforms in comparison to them by a large margin. This answers the question *How can ensembling help in better detecting duplicate questions from a small dataset?*

ID	Question 1	Question 2	Label
1	Is Deadpool overrated?	Why is Deadpool overrated?	Duplicate
2	Why do people believe in karma?	Why do some people not believe in Karma?	Non-Duplicate

Table 4.5: Example of questions mislabeled by Sentence Interaction models. The sentences have high overlap, which makes it harder for the models to understand the differences. The questions were correctly labeled by the Sentence Encoding model.

ID	Question 1	Question 2	Label
1	Is it possible that Donald Trump has a mental problem ?	Is Donald Trump mentally stable ?	Duplicate
2	Why does it rain so heavy ?	Why does the road become bad after a heavy rain ?	Non-Duplicate

Table 4.6: Example of questions mislabeled by the Sentence Encoding model. In this case the model does not perceive the different contexts and synonyms. The questions were correctly labeled by Sentence Interaction models.

4.6 Summary

In this section we have introduced the problems we identified in the study of DQD and proposed a series of solutions with which we attempt to solve them. We focus on small datasets and aim to find appropriate text representation, data augmentation, transfer learning and ensembling techniques to further advance the study of DQD.

5 Evaluation

This chapter describes the implementation of the experimental framework and the experiments conducted to evaluate the solutions proposed in Chapter 4. We begin with describing the datasets and then elaborate on the implementation and considerations made during this step. Our main focus is on advancing duplicate question detection in the context of limited training datasets. We present the setup of our experiments and analyze their results, commencing a discussion on how they fit in the current research of the NLP community. We run several experiments, each one addressing one of the proposed solutions, with the goal to study the contribution of each one for the final classification results. We will use two different datasets for identifying duplicate questions from which we sample smaller datasets. We use F1-score as evaluation metric. We also present the tools that we used in the development of our framework.

5.1 Datasets

We use two publicly available datasets for our analysis: Quora[1] and AskUbuntu [2]. Both datasets have a similar format, with a pair of questions and a binary label denoting whether they are duplicate or non-duplicate. The datasets have different characteristics, and one of the important contrasts stems from their vocabulary: Quora is a general-purpose dataset, with questions asked about a diverse range of fields, while AskUbuntu is a dataset on technical questions. To better understand what they have in common and how they diverge, we conduct a data analysis on both datasets, which we present in Sections 5.1.1 and 5.1.2.

5.1.1 Quora

The Quora question pair dataset was released in 2017 to enable research on semantic equivalence. The dataset contains over 404, 000 pairs of English questions on different subjects such as technology, culture, philosophy and sometimes contains mathematical symbols and special characters from other languages. The questions were annotated by human experts. So the labels are expected to be subjective and might not be 100% accurate. Nonetheless, it is not clear how the labeling process was organized, in terms of how many experts participated and how the final label was chosen.

We present an overview of the main properties of the dataset from a high-level analysis. The dataset presents imbalance, having more non-duplicates: only 37% of the questions

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[2]https://askubuntu.com/

are labeled as duplicate. In addition, approximatively 67% of the total amount of questions are unique. On the other hand, some of the questions are paired to themselves, and while looking at these questions we see the first instance of noise present in the dataset: we find a small number of questions which even though have the same text, they are labeled as non-duplicate as in Table 5.1. As there are only a few pairs like this, the training process should not be affected.

ID	Question 1	Question 2	Label
1	Do you think I have OCD	Do you think I have OCD?	Non-Duplicate
2	What is the answer to this "IQ" test question?	What is the answer to this IQ test question?	Non-Duplicate

Table 5.1: Example of pairs of questions that have the same text but are labeled as non-duplicate.

In terms of vocabulary, the dataset contains 95,595 unique words. On average, the sentences have about 60 characters with a standard deviation of 31 characters. These characters add up to approximatively 11 words per question. The sentence length in terms of words is highly skewed to the right, with some sentences reaching over 200 words, as we can see in Figure 5.1. The questions contain approximatively 4.22 stop words (i.e. most common words in a language, such as "the", "a", "and", "but" etc.) and most of them, as we expected, begin in an interrogative way with one of the words: "What", "How", "Why", "Is", "Which", "Can" and "Who".

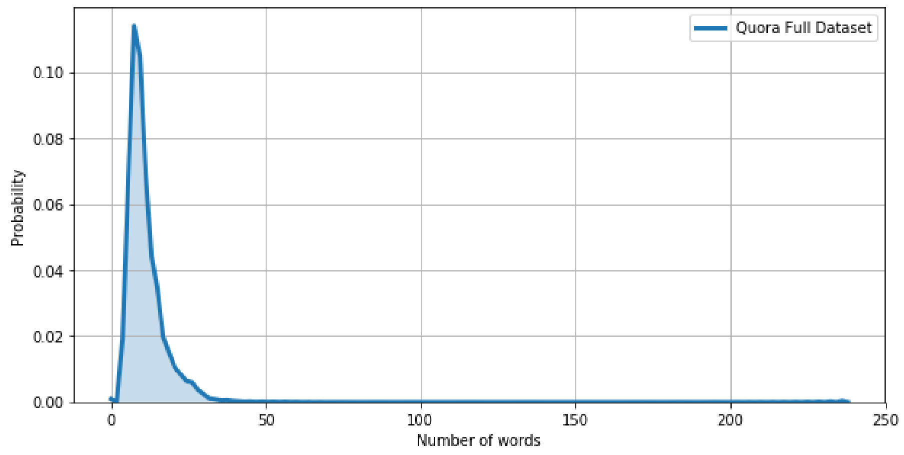


Figure 5.1: Histogram representing the number of words in Quora dataset. The mean number of words is 11.06, with a standard deviation of 5.89. The maximum number of words is 237.

We measure the mean lexical overlap by lemmatizing each pair of questions and finding the ratio between the number of unique words in common and the total amount of unique words. In general, the pairs in this dataset have a high overlap of 45%, which indicates that many questions are not complex paraphrases, but instead fit in the following categories from Section 1.2:

- **Morphology changes**: "What does manipulation mean?" and "What does manipulation means?"
- **Structural changes**: "Why is creativity important?" and "Why creativity is important?"
- **Lexicon changes**: "What can make Physics easy to learn?" and "How can you make physics easy to learn?"

On average, duplicate pairs have an overlap of 56% while non-duplicates overlap by 36%. Figure 5.2 shows that questions with very low overlap are more likely to be non-duplicate, while both duplicates and non-duplicates can have high overlap.

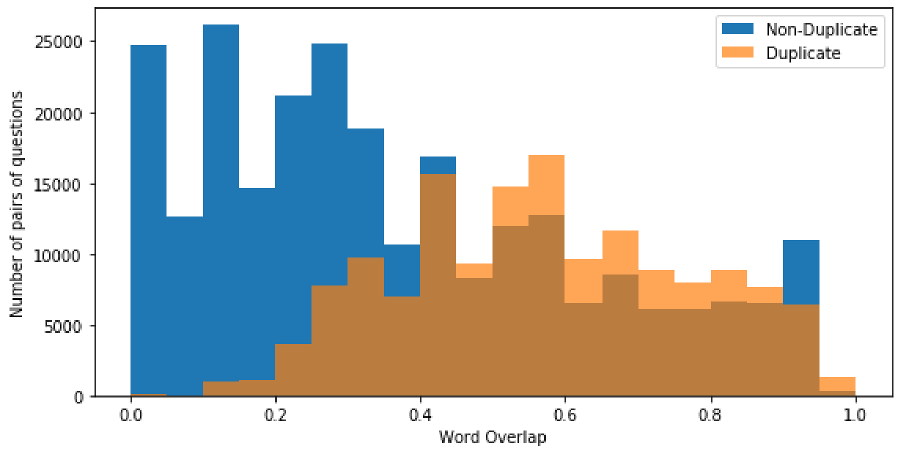


Figure 5.2: Histogram representing the label distribution according to word overlap score for Quora dataset. The brown region is where the two distributions overlap.

To make our results comparable to the literature, we used the data split created and released by the authors of BiMPM model [Wang et al., 2017]. The development set and the test set each contain a balanced sample of 10,000 pairs of questions and the remaining questions are used as the training set. From this point on, when we talk about Quora training, dev or test set we refer to this dataset partition. We will specifically refer to this training set as Quora Full.

The focus of our research is on detecting duplicate questions by training on smaller sample sizes. For this reason, we randomly sample six datasets from the training set. We will refer to these training sets as Quora 5k, Quora 10k, Quora 15k, Quora 20k, Quora 25k, Quora 30k (based on how many pairs of questions are contained in each).

5.1.2 AskUbuntu

This section introduces the main characteristics of our second dataset, to which we will refer as AskUbuntu. AskUbuntu[3] is one of the Stack Exchange [4] sub-forums, addressed to the community of Ubuntu operating system. This dataset was released as part of the reproduction study conducted by Silva et al. on the work of Bogdanova et al. ([Silva et al., 2018], [Bogdanova et al., 2015]). It contains in total almost 31,000 labeled pairs of questions, thus being a lot smaller than the original Quora dataset. The dataset was obtained after preprocessing the Stack Exchange data dump[5] from September 2014. The authors provide a script for preparing the dataset by removing images, URLs and code snippets. On all Stack Exchange forums, moderators are able to mark a new question as duplicate of an older one and after five similar votes the question is officially marked as duplicate and linked to the older one. Thus, duplicate questions in the data dump are linked, enabling the authors to label them accordingly.

AskUnbuntu is designed to be balanced, with 50% duplicate questions and 50% non-duplicate questions. In this dataset approximatively 80% of the questions appear only once. A question is typically formed from a title and body as shown in Table 5.2.

Question 1	Question 2
Title: How do I install Ubuntu?	**Title**: How can I install 12.04.2?
Body:I would like to see a full how-to on how to install Ubuntu.	**Body**: I have never installed an OS before. I upgraded to 13.04 and it has been a total disaster. I now want to go back to 12.04.2, but will have to install it totally. I tried following the instructions, but the computer doesn't do anything that instructions tell me to do. I have downloaded it to a flash drive, but there seems to be no starting point that I can identify. Where do I start? Also, is it possible to download it directly from the hard drive, or does it have to be from an external source? Thank you.

Table 5.2: Example of questions marked as duplicate in AskUbuntu dataset.

Questions in this dataset tend to be a lot longer, as some users typically use the body of the questions to thoroughly explain their query. Figure 5.3 shows that the word count distribution is highly skewed to the left with questions having on average 87 words and

[3]https://askubuntu.com/
[4]https://stackexchange.com/
[5]https://archive.org/details/stackexchange

standard deviation of 76 words. This indicates that the length of the texts varies a lot. With a closer look, we see that the majority of the questions have between 30 and 80 words. On average, the questions can have up to 41 stop words. This leads us to believe that the technical problem that a user is dealing with can be determined based on just a few keywords.

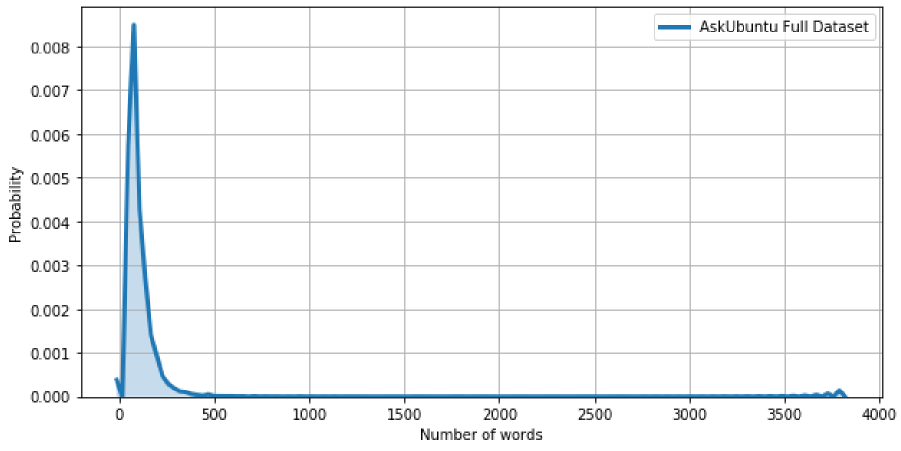


Figure 5.3: Histogram representing the number of words in AskUbuntu dataset. The mean number of words is 87.52, with a standard deviation of 76.08. The maximum number of words is 3804.

In terms of vocabulary, the dataset contains 52, 067 unique words. As expected, many terms are technical, such as ”Ubuntu”, ”Windows”, ”install”, ”boot”, ”error”, ”USB”, ”sudo”. Questions in this dataset tend to begin with ”How”, ”What”, ”Ubuntu”, ”Is”, ”Can”, ”Why”, ”My” which is slightly different from how users of Quora behave. Furthermore, the titles of the questions are not always formulated as questions, sometimes they are a short summary of the problem the user is trying to solve (”Unable to install java on my ubuntu”), followed by a question in the body of the query.

We compute lexical word overlap and plot the distribution of labels according to this score. Figure 5.4 shows that the pairs of questions in this dataset have little overlap, irrespective of their label. Duplicate questions tend to have more overlap (on average 27%) as compared to non-duplicates (on average 19%), but overall the pairs display few words in common. This shows us that questions from this dataset are complex paraphrases, where users express the same meaning with different words. One factor that we need to consider in this case is that typically in technical Q&A website users might have slightly different problems, which can be solved with the same fix. This means that it is not surprising that the questions have less overlap.

Silva et al. provide a train/dev/test data split which we also use[Silva et al., 2018]. The training set has 24, 000 pairs of questions, while development set has 1, 000 and test set has 6, 000. We sample four smaller datasets from the training set such that they have approximatively 37% duplicates each. We make the dataset imbalanced so that

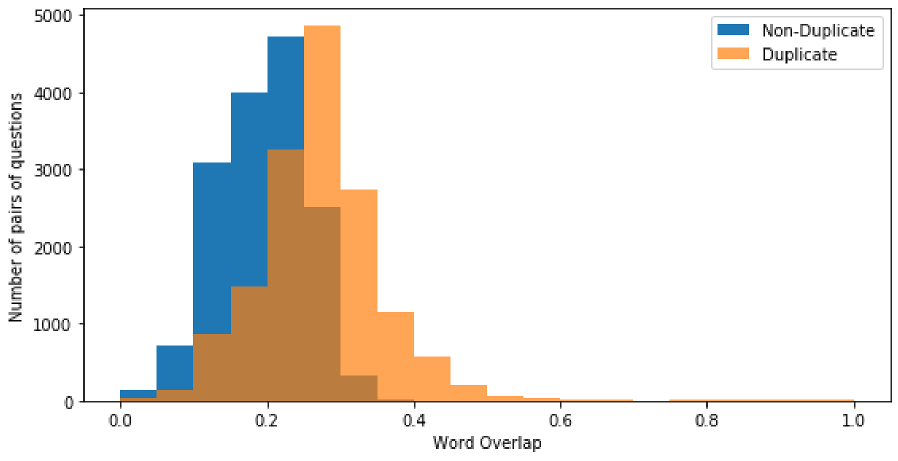


Figure 5.4: Histogram representing the label distribution according to word overlap score for AskUbuntu dataset. The brown region is where the two distributions overlap.

it better reflects a real-life scenario. We will denote these datasets as AskUbuntu 5k, AskUbuntu 10k, AskUbuntu 15k, AskUbuntu 19k, based on how many pairs of questions each contains.

Our deep learning models and techniques behave differently on these two datasets as they have quite different characteristics (see Section 5.3). Our assumptions is that the models perform better with the longer texts from AskUbuntu, because they are able to extract more meaning from the key words. Furthermore, AskUbuntu has a richer vocabulary than Quora, having more than half the number of unique words that Quora has, even though it is 15 times smaller. We summarize the different properties of the datasets in Table 5.3.

5.2 Framework Implementation

This section describes the general setup of the framework that was implemented with the goal of advancing the study of duplicate question detection. We also present the assumptions and considerations that we take into account. The code for the experiments is available at: `https://github.com/andra-pumnea/Thesis`.

5.2.1 Preprocessing

In this section we discuss the steps we use for transforming the datasets into input suitable for neural networks. Both datasets have the same scheme, consisting of question1, question2 and label. Preprocessing begins with reading the text for question 1 and question 2 from the file, along with the label (1 for duplicate, 0 for non-duplicate). There are different ways in which text can be preprocessed before being fed to an algorithm,

Dataset	Vocabulary size	Avg Length (words)	Word Overlap	Train	Dev	Test
Quora 5k	10,436	12.78	52.6%	5k	10k	10k
Quora 10k	25,145	12.70	52.7%	10k	10k	10k
Quora 15k	18,421	12.70	52.8%	15k	10k	10k
Quora 20k	21,531	12.73	52.8%	20k	10k	10k
Quora 25k	24,292	12.74	52.6%	25k	10k	10k
Quora 30k	26,536	12.73	52.6%	30k	10k	10k
Quora Full	95,595	11.06	45%	380k	10k	10k
AskUbuntu 5k	20,912	89.82	22.5%	5k	1k	6k
AskUbuntu 10k	30,387	89.70	22.2%	10k	1k	6k
AskUbuntu 15k	37,589	89.90	22.2%	15k	1k	6k
AskUbuntu 20k	42,539	89.79	22.2%	19k	1k	6k
AskUbuntu Full	52,067	87.52	23%	24k	1k	6k

Table 5.3: General descriptions of the datasets used in this work.

which mainly consists of removing special characters, lowercasing, removing stop words, generating n-grams, lemmatization and stemming. The main purpose of these steps is to remove noise i.e. elements that do not add meaning to sentences.

We have opted for removing all special characters [! ” # % & ( ) * + , - . / : ; <= >? @ [ ] ˆ_ { }] as they do not contribute to the context of a sentence. For example, some simple counts on Quora dataset showed us that 99.87% of questions contain the question mark symbol, while other symbols such as full stop appear 6.39% and a small number of questions (less than 1%) contain math and programming symbols such as <, >, = etc. We also lowercase all the text.

We choose not to remove stop words as this might alter the meaning of a sentence. For example, the relationship between the duplicates in Table 5.4 becomes ambiguous after removing stop words with the *NLTK* toolkit [6]:

	Original	Stop words removed
Q1.	Why do people not like Donald trump running for president?	Why, people, like, Donald, trump, running, president, ?
Q2.	Why are people so against Donald Trump running for president ?	Why, people, Donald, Trump, running, president, ?

Table 5.4: Examples of sentences losing initial meaning after stop word removal.

Lemmatization is the process of returning a word to its base or dictionary form, while stemming removes the suffix of a word. As we want to retain as much as possible of the initial meaning of a sentence, we do not apply neither of these techniques to our datasets.

[6]https://www.nltk.org/

None of the works described in Section 3.1 utilize this step either and we consider it is for a good reason.

After the cleaning step, each sentence goes through a tokenization process, in which it is split in individual words based on blank spaces. We create a dictionary (also known as word index) in which each unique word is assigned an index. This dictionary is used to replace every word in a sentence with its index, thus obtaining word vectors. For example, the question becomes the vector:

Input: [how, do, i, get, funding, for, my, web, based, startup, idea]
Output:$[4, 9, 5, 30, 1655, 14, 17, 259, 505, 546, 429]$

The same dictionary can be used for mapping the vectors back to sentences.

The embedding layer requires that all vectors have the same length, which is initially not the case as sentences naturally vary in length. There are two ways to obtain such vectors: *truncating* sentences that are too long and *padding* sentences that are too short. Sequence length is a parameter to be determined. [Homma et al., 2017] suggests picking a sequence length of 30 words for Quora dataset, such that more information can be captured from longer sequences. Instead, we chose a sequence length of 40 words per sentence, mainly because we wanted to truncate as few sentences as possible in Quora dataset. As AskUbuntu sentences are longer, we chose a sequence length of 60 words. When a question has fewer than 40, respectively 60 words, it is *padded* with 0, which means that the rest of the vector is filled up with 0, either before the word indices or after. A smaller sequence size would help reduce training time. Continuing the example above, after padding it would look like this:

Input: $[4, 9, 5, 30, 1655, 14, 17, 259, 505, 546, 429]$
Output:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 9, 5, 30, 1655, 14, 17, 259, 505, 546, 429,]

These resulting vectors are passed as input to the neural network.

5.2.2 Deep Learning Models

This section describes the models used in our work together with their implementation details and the configuration of the baseline. The experimental framework contains three deep learning models, which were previously outlined in Section 3.1, namely: Siamese GRU [Homma et al., 2017], DecAtt [Parikh et al., 2016] and ESIM [Chen et al., 2016]. On the high-level the models are similar: they take word embeddings as input, then they generate a representation for the questions which is used for outputting a label. Their main difference stands in the way the representation is obtained. The goal of our first experiment that we perform is to establish the baseline and we will refer to it as *Experiment 1: Baseline configuration.* Results of this experiment are presented in Section 5.3.1.

We replicate the **Siamese GRU** model based on the original paper, aiming to reproduce the authors' results on the *GRU, 2-layer similarity network*. We encode each sentence using a shared GRU layer and obtain h_1 and h_2 representations. Then, the following concatenated vector v is fed to the two-layer feedforward network:

$$v = [h_1 \ h_2 \ (h_1 - h_2)^2 \ h_1 \odot h_2]$$

In this vector we concatenate the encoded representations h_1 and h_2, their squared difference and their dot product. We set the hidden layer size to 250 neurons as suggested in the paper and add a dropout of 0.01 between the dense layers, to reduce overfitting. Our best results on Quora Full dataset is 83.33% accuracy, as compared to 84.95% reported in the paper. The small difference might stem from the different train/dev/test split that the authors used: they used the same test set, but split the remaining non-test data into 85% training and 15% development set.

The **DecAtt** and **ESIM** models were used by the winning team in the *Kaggle Quora Question Pairs Competition*[7]. The winning team released the code[8] of their interpretation of the papers, which we used in our framework. In their interpretation DecAtt has a small variation, which includes performing subtraction when obtaining the representation of the questions, which we keep. The models are already tuned to the problem of DQD, so we did not do any further hyperparameter tuning. For evaluating the performance of the models, we trained and tested them on the original dataset (SNLI [Bowman et al., 2015]). DecAtt achieved an accuracy of 84.9%, as compared to 86.3% in the original paper, while ESIM achieved 89.1% versus 88%. The reproduction of Lan et al. obtained 85.6%, respectively 87% [Lan and Xu, 2018] on the same dataset, so we consider our models' performance acceptable, considering they are already tuned to Quora dataset. Finally, we compare the results our models obtain on Quora dataset to the ones obtained by the reproduction mentioned above: we obtain 85.32% versus their 84.5% for DecAtt and 85.91% versus 85% for ESIM. This further confirms that our models are working as expected. The differences might stem from the fact that we used different deep learning toolkits, as we used Keras with Tensorflow backend, while the other papers reported Pytorch or Theano. In addition, neural networks are stochastic, so they are non-deterministic, which means that identical results would be very hard to achieve.

The networks are trained with Adam [Kinga and Adam, 2015] optimizer and a learning rate of 0.001 to minimize binary cross-entropy loss. We use GloVe embeddings, with 300 dimensions for our initial tests, in order to assess the quality of the implementations and have a fair comparison with the original work. The input is processed in batches of 50. We train the networks for at most 50 epochs and use the *Early Stopping* technique (see Section 2.2.4). If the network does not improve over the course of three epochs, it automatically stops before reaching 50 epochs.

The baseline against which we will compare the different strategies will consist of these three neural networks and will use GloVe embeddings for the embedding layer.

[7]https://www.kaggle.com/c/quora-question-pairs
[8]https://www.kaggle.com/lamdang/dl-models

Depending on the complexity of the network and the size of the dataset, each network takes a different amount of time to train. On average, on the full Quora dataset Siamese GRU takes approximatively 2 hours to train, DecAtt takes approximatively 50 min and ESIM takes approximatively 12 hours. Table 5.5 depicts how we configure the baseline.

Setup	
Model	- SiameseGRU - DecAtt - ESIM
Dataset	- Quora (5k, 10k, 15k, 20k, 25k, 30k) - AskUbuntu (5k, 10k, 15k, 19k)
Word embedding	GloVe

Table 5.5: Experiment 1: Baseline configuration.

5.2.3 Word and Sentence Embeddings

This section describes the different strategies for generating and using embeddings. In our approach we use both word embeddings and sentence embeddings. We will refer to the experiment that evaluates this strategy as *Experiment 2: Effect of pre-trained embeddings* and present its results in Section 5.3.2. The sentence embeddings will be incorporated in the neural networks as an additional feature to the word embeddings.

For a deep learning model to be able to learn from text data we need to create a word embedding matrix. In order to obtain this matrix we use the GloVe and FastText (see Sections 2.4.3 and 2.4.4) pre-trained word vectors, which we load into memory. We explain how to obtain the word embedding matrix from GloVe vectors, with a small difference for FastText.

The embedding matrix, which will be passed to the embedding layer, requires two parameters: the number of unique words it should contain and the dimension of the vectors. As our largest dataset is Quora (whose training set has in total approximatively $85,000$ unique words), we decided on picking the $80,000$ most frequent words to add to the embedding matrix. In case the dataset has less than $80,000$, we initialize the rest of the matrix with zero vectors in order to be able to perform transfer learning (see Section 5.2.5). The dimension of each word vector will be 300.

We retrieve the word vector from GloVe file and add it to the matrix based on the index assigned in the dictionary to that word. The words which were not found in the word embedding file are initialized to a vector of zeros. This is not ideal, but the alternative would be to initialize the vector with random numbers, which might give unwanted meaning to that word in the vector space. In contrast, when we compute the embeddings with FastText, in case the word does not have any known embeddings, it is decomposed in n-grams. The embeddings for respective n-grams are retrieved and added to the embedding matrix. Only in the case when the n-grams do not have any embeddings, the word vector is left as 0.

For sentence embeddings we use the Universal Sentence Encoder (USE) (see Section 2.4.5). The model is provided through *tensorflow_hub* library and requires that the sentences are fed to it in the form of strings, no tokenization necessary. After passing through this model, each sentence will be represented by a 512 dimensional vector. Each new sentence vector goes through a fully connected layer with 256 neurons, reducing the size of the initial sentence vector to half. We compute cosine distance as a measure of similarity between the sentence vectors and finally concatenate the distance vector to the representation of the questions obtained after the transformations that each model applies to the word embeddings. Initially, we tried computing the cosine distance between the two 512 dimensional vectors, but using a fully connected layer yielded better results and proved faster.

We evaluate the behavior of the different word embeddings on our datasets and subsamples, as well as their combination with information from sentence embeddings. The papers cited in Section 3.1 report using GloVe as their embedding of choice, so we aim to explore whether newer approaches address the limitations of GloVe. Table 5.6 summarizes the setup of this experiment.

Setup	
Model	- SiameseGRU - DecAtt - ESIM
Dataset	- Quora (5k, 10k, 15k, 20k, 25k, 30k) - AskUbuntu (5k, 10k, 15k, 19k)
Word Embeddings	- GloVe - GloVE+USE - FastText - FastText+USE

Table 5.6: Experiment 2: Effect of embeddings.

5.2.4 Data Augmentation

We refer to the experiment that evaluates data augmentation techniques as *Experiment 3: Effect of Data Augmentation* and report the results in Section 5.3.3.

First of all, we perform a simple form of what is called *word embedding perturbation* [Zhang and Yang, 2018]. As explained in Section 4.3, we add a small amount of noise sampled from the Gaussian distribution to the word embedding layer. We experiment with three values of noise $\{0.01, 0.05, 0.1\}$. We apply Gaussian noise directly on the embedding layer and for this we use the Gaussian Layer provided by Keras [9].

Second of all, we generate new question pairs using three methods: reversing the order of questions, replacing words with their synonyms in each question and selecting

[9]https://keras.io/

a number of pairs of questions based on a heuristic and adding them in reverse order. An overview of these techniques was presented in Section 4.3, here we discuss the technicalities involved in applying these methods.

The first two methods would effectively double the initial dataset, maintaining the initial distribution of labels. For the third method we sample the training set based on the score of word overlap which we described in Section 5.2.3. With this method, the label distribution changes slightly in favor of duplicate questions, based on the characteristics of the corresponding training set. We extract all questions that have word overlap higher than 0.6 and all questions that are duplicate but have at most 0.4 word overlap. The number of questions extracted for each dataset is presented in Table 5.7. As expected, AskUbuntu dataset has a lot less questions with high overlap compared to Quora. On the other hand, it has more questions that have low overlap and are labeled as duplicate (i.e. complex paraphrases).

Dataset	Initial size	Questions with overlap >0.6	Duplicates with overlap <0.4	Total
Quora 5k	5,000	1,475	820	7,295
Quora 10k	10,000	2,973	649	13,622
Quora 15k	15,000	4,452	1,614	21,066
AskUbuntu 5k	5,000	18	1,764	6,782
AskUbuntu 10k	10,000	28	3,541	13,569
AskUbuntu 15k	15,000	40	3,535	18,575

Table 5.7: General description of the datasets augmented based on heuristics method.

We evaluate the behavior of our models with these new enhanced training datasets. In order to be able to compare to our baseline, we opt for GloVe embeddings and keep them fixed through the experiment. Table 5.8 depicts the setting of this experiment.

Setup	
Model	- SiameseGRU - DecAtt - ESIM
Dataset	- Quora (5k, 10k, 15k) - AskUbuntu (5k, 10k, 15k)
Word embedding	GloVe
Data Augmentation	- Gaussian Noise = $\{0.01, 0.05, 0.1\}$ - Reverse-based - Thesaurus-based - Heuristic-based

Table 5.8: Experiment 3: Effect of data augmentation.

5.2.5 Transfer Learning

Two experiments emerge in the transfer learning setup: *Experiment 4: Effect of entire model transfer* and *Experiment 5: Effect of pre-training and fine-tuning.* We apply transfer learning in two different ways: first method consists of fully training and tuning a model on a dataset and evaluating on a different dataset; second method corresponds to pre-training the layers of the model on a dataset and fine-tuning and evaluating it on a new dataset.

The first approach consists of simply training and tuning all the parameters of a model on a dataset and testing it on a different dataset. In this setting, we simply load the previously trained model and evaluate it on the test set of the target dataset. This way the results are directly comparable with training the model entirely on the target dataset. In this context, we are interested in how the models behave in two different situations:

- **When dealing with the same task, but a different word distribution and domain**. In this case, we train the model on the bigger Quora dataset (Quora Full) and evaluate it on the smaller AskUbuntu samples. AskUbuntu is a more domain specific dataset, encompassing a wide range of technical terms (see Section 5.1.2), while Quora is more noisy and general (see Section 5.1.1). As the relationship between pairs from both datasets are the same (duplicate, non-duplicate), it is relevant to observe how semantics related knowledge is transferred between them.

- **When dealing with a different, but somehow related task**. In this case we train the models on the SNLI dataset and test it on Quora test set. As explained in Section 3.2, NLI is related to DQD because a paraphrase relationship can be considered a symmetrical entailment relationship. We have chosen the SNLI dataset on the basis of the research conducted by Conneau et al.: they claim this dataset is useful when learning universal embeddings because of its complex semantics and high amount of labeled data [Conneau et al., 2017]. Thus, we are interested in how our models can generalize on the problem of DQD from learning this related task.

The second approach is similar to the first one. We load the previously trained models and experiment with not freezing the layers and freezing the weights of all the layers, except the last layer, the one that performs the classification. Hence, the model goes through another training process, where it is tuned on the target dataset. When the layers are frozen (meaning they remain unchanged during this process), only the weights in the last layer are updated. Otherwise, the weights on all the layers are updated. This way, the model leverages the knowledge learned from the previous dataset and does not begin from scratch. We will evaluate this technique in the same two settings discussed above: we tune the model trained on SNLI dataset on Quora samples and the model trained on Quora Full on AskUbutnu samples. Similarly to the other experiments, we use GloVe embeddings and all our models. Table 5.9 summarizes this setup.

Setup	
Model	- SiameseGRU - DecAtt - ESIM
Dataset	- Quora (5k, 10k, 15k, 20k, 25k, 30k) - AskUbuntu (5k, 10k, 15k, 19k)
Word embedding	GloVe
Transfer Learning	- Quora+AskUbuntu (test set) - SNLI+Quora (test set) - SNLI+ Quora (5k, 10k, 15k, 20k, 25k, 30k) - Quora+AskUbuntu (5k, 10k, 15k, 19k)

Table 5.9: Experiment 5: Effect of transfer learning. By transfer learning we refer both to entire model transfer and pre-training and fine-tuning.

5.2.6 Ensembling

The experiment which evaluates ensembling is called *Experiment 6: Effect of ensembling.* As discussed in Secion 3.1, there are two main types of approaches when it comes to DQD (sentence encoding and sentence interaction) each with its own strength [Yu et al., 2018]: learning separate representations for the questions and then combining them, respectively, modeling their interaction and making an abstraction on this output. As we have included both types of models in our framework, we are interested in leveraging the different perspectives that they tackle and resort to **stacking** them.

In this approach we create a new model that combines the predictions of our models. We perform simple averaging of the outputs of the models involved in the ensemble and expect that this will work better than any individual model. For each neural network we will load the saved weights of the best performing model. This type of ensembling does not need training because performing the averaging operation does not require any parameters. On the other hand, this leads to increased prediction and evaluation times, which can be an impediment in a real-life scenario.

To evaluate the performance of ensembling we will stack the three models together and assess them on our chosen datasets. Furthermore, we will also experiment with pairs of models and check how their prediction power behaves in this setting. The setup is summarized in Table 5.10.

5.2.7 Tools

The implementation of the framework and data analysis scripts were done in Python 3.6 programming language. For the deep learning models we used Keras [Chollet, 2015] library with Tensorflow backend. Table 5.11 describes the infrastructure used for training the neural network and running the experiments.

Setup	
Model	- SiameseGRU - DecAtt - ESIM
Dataset	- Quora (5k, 10k, 15k, 20k, 25k, 30k) - AskUbuntu (5k, 10k, 15k, 19k)
Word embedding	GloVe
Ensemble	- SiameseGRU+DecAtt - SiameseGRU+ESIM - DectATT+ESIM - SiameseGRU+DecAtt+ESIM

Table 5.10: Experiment 6: Effect of ensembling.

Operating System	**RAM**	**GPU**
Ubuntu 16.04	4 GB	NVIDIA GTX 980

Table 5.11: Hardware configuration for the GPU server. All the experiments using neural networks were deployed using this configuration.

5.3 Results and Discussion

In this section we present and discuss our main findings from each experiment. First we present the results of the baseline, followed by discussing how each component (text representation, data augmentation, transfer learning and ensembling) affects it.

All the results that we present are in terms of F1-score. We chose this metric because of the imbalance that the datasets present, as this would provide a more clear estimate of the performance of the models and proposed components. For each experiment the model is run three times and we report the average of F1-score. The reason behind this is the randomness in neural networks, which leads to slightly different results every time.

5.3.1 Baseline Results

This section presents the results of *Experiment 1: Baseline Configuration.* In Table 5.12 we report the F1-scores of our three deep learning models after training on the different subsamples of Quora dataset. We can see that all of them give reasonable results in this setting. The two sentence interaction models (DecAtt and ESIM) achieve a high F1-score throughout the entire experiment, while SiameseGRU considerably under-performs in comparison to them (the score is up to 10% lower). A possible explanation for this is that SiameseGRU is more sensitive to the noise present in the dataset (and would need more data in order to generalize better), while DecAtt and ESIM are more robust because of the attention mechanism that helps them focus on the keywords and identify subphrases that are similar. We see that as we increase the random samples by $5,000$ questions the results increase by approximatively 7% for SiameseGRU and approximatively 6% for

DecAtt and ESIM. Another interesting aspect is that even though ESIM uses BiLSTM to better capture information in the two sentences (instead of a simple feedforward network like in DecAtt), this does not change much in the overall performace of the model on Quora dataset. On the other hand, DecAtt is more desirable to be used because of its training speed.

Quora

Model	5k	10k	15k	20k	25k	30k
SiameseGRU	0.6201	0.6421	0.6664	0.6778	0.6758	0.6985
DecAtt	0.7145	0.7454	0.7732	0.7728	0.7691	0.7893
ESIM	0.7229	0.728	0.7418	0.774	0.7729	0.7833

Table 5.12: Results from running the baselines on Quora subsamples ranging from 5,000 pairs of questions to 30,000 pairs of questions.

In Table 5.13 we show the results on AskUbuntu dataset. The models show the same trend as on Quora dataset: DecAtt and ESIM outperform SiameseGRU by a large margin. Interestingly, the F1-scores are considerably higher compared to Quora dataset. We believe there are two reasons for this: first, the question length is higher (the majority of questions have 60 words each, while Quora has on average 11 words, up to a maximum 40 words) which makes it easier for the models to capture more meaning from the text and extract essential features; second, the labels in the dataset are less noisy (we explained the labeling procedure in Section 5.1.2). Furthermore, the fact that there is less word overlap between the pairs of questions seems to be beneficial, because in this case the sentence interaction models do not need to find the small details that are different (like in the case of Quora duplicate questions). On the other hand, we notice that the training with DecAtt on the AskUbuntu samples is less stable than the other two models and there is higher variation between the three runs, with standard deviation up to 5% as compared to at most 2%.

In order to better understand the results, we experiment with different question lengths, as this is one of the main factors that differentiates between our datasets. We pick the AskUbuntu dataset with $5,000$ questions and we experiment with question lengths ranging from 10 words to 60 words, increasing in steps of 10. Figure 5.5 depicts the results of this experiment. When the questions have lengths of 10 (similarly to Quora dataset which on average has 11 words per question), DecAtt reaches an F1-score of approximatively 70.76% (which is close to the performance of Quora 5k). On the other hand, as the question length increases, the results of the network show much higher standard deviation. One possible reason behind this is exploding gradient (see Section 2.3.1), which can make the network unable to learn long-term temporal relations. We hypothesize that the use of BiLSTM helps stabilize ESIM as compared to the feedforward network used by DecAtt, which is why ESIM gives more stable results throughout the training process. Based on this observation, we decide not to use DecAtt in our evaluation for AskUbuntu dataset, as the results are not reliable enough.

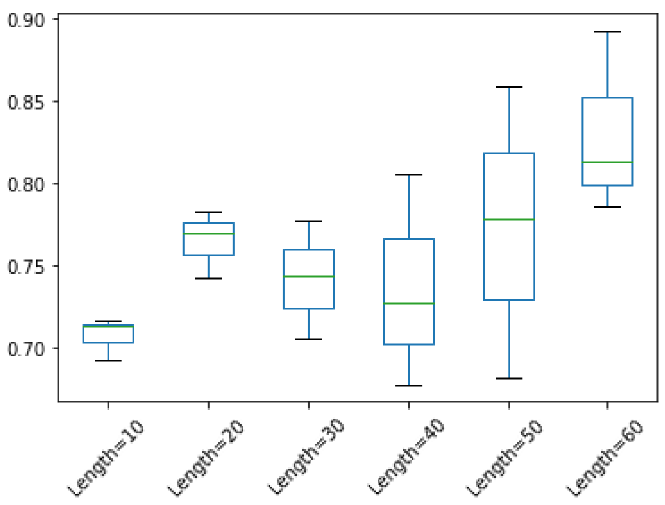


Figure 5.5: Boxplot showing the behaviour of DecAtt on the AskUbuntu 5k dataset, with varying question length. We can notice that as the question length increases, the results of the network becomes less stable.

AkUbuntu

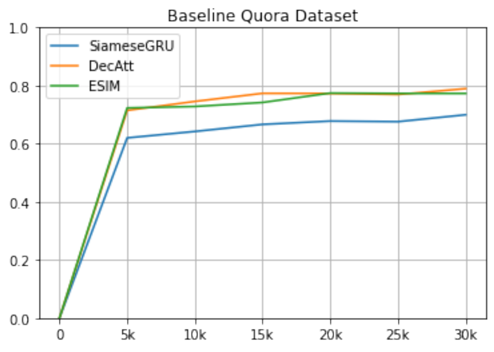
Model	5k	10k	15k	20k
SiameseGRU	0.7155	0.7328	0.7290	0.7590
DecAtt	0.8476	0.813	0.7864	0.8928
ESIM	0.8727	0.8946	0.9024	0.9089

Table 5.13: Results from running the baselines on Askbuntu subsamples ranging from 5,000 pairs of questions to 20,000 pairs of questions.

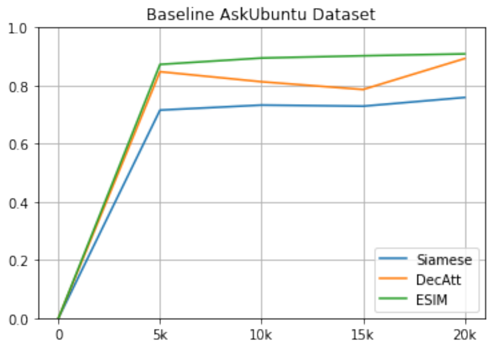
Figure 5.6 visualizes the evolution of the results with the increase of training set size. The curves of the sentence interaction models are almost parallel to SiameseGRU. From this section, we can conclude that the content of the questions is less important for performing duplicate question detection, as the models are able to deal well with technical terms in AskUbuntu dataset. Factors such as size of the dataset, level of noise (both in labels and in text) and question length contribute more to the classification results. Hoogeveen reaches a similar conclusion in her [Hoogeveen, 2018].

5.3.2 Effect of text representation

In thise section we discuss how each each model is affected by the different text representations. Every question is represented as discussed in Section 4.2, either with Glove or FastText pre-trained embeddings and an additional feature of sentence embeddings from Universal Sentence Encoder (USE). In Table 5.14, we present the number of unique words contained in each Quora sample, along with the number of words not represented



(a) Quora Baseline.



(b) AskUbuntu Baseline.

Figure 5.6: Experiment 1: Baseline results.

in Glove and FastText. On average, 6.4% of the vocabulary of the samples is not contained in GloVe embeddings and 6.9% is not contained in FastText. With GloVe, these embeddings are initialized to a zero vector while with FastText they are initialized to the sum of their pre-trained n-grams.

Dataset	5k	10k	15k	20k	25k	30k
Words in index	10,436	15,228	18,421	21,531	24,292	26,536
Glove Null Embeddings	421	775	1,124	1,464	1,895	2,287
FastText Null Embeddings	458	836	1,186	1,631	2,039	2,420

Table 5.14: Count of words in index and null embeddings for each sample from **Quora** samples.

Table 5.15 presents the number of unique words for AskUbuntu samples, as well as the number of words that were not found in GloVe or FastText pre-trained embeddings. On average 27.6% of the vocabulary of this dataset is not found in GloVe and 31.01% is not found in FastText. We can see that the number of words without a pre-trained embedding is much higher in comparison with Quora samples. The kind of words that neither of text representations have are about libraries, hardware components and other technical terms (for example "lib80211", "guake", "f9d527c70255"). In this case, GloVe helps with filtering unnecessary descriptions and misspellings, likely clearing noise. On the other hand, FastText will create an embedding for these words based on their n-grams, which in some cases might be useful.

The results of this experiment on Quora samples are illustrated in Figure 5.7 and they show that the different text representations perform similarly. The results indicate that the siamese network benefits from FastText and USE as the size of the dataset increases. In fact, on the smallest sample (Quora 5k), the baseline using GloVe is clearly outperformed by all other embeddings. This behavior can be explained by the fact that

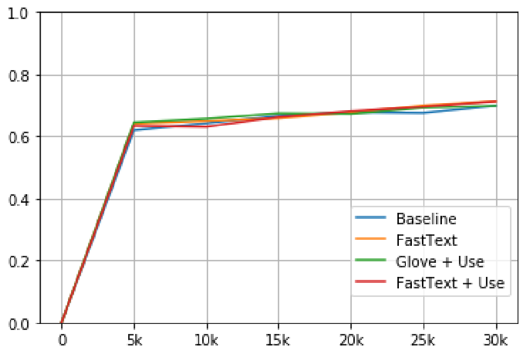
Dataset	5k	10k	15k	19k
Words in index	20,912	30,387	37,589	42,539
Glove Null Embeddings	4,541	8,010	11,367	13,709
FastText Null Embeddings	5,074	9,082	12,730	15,322

Table 5.15: Count of words in index and null embeddings for each sample from **AskUbuntu** samples.

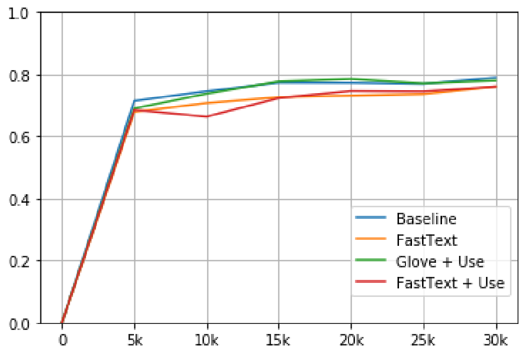
the siamese network needs more data in order to generalize better and thus benefits from having embeddings for all words in the sample, unlike when using GloVe. The results also show that adding an extra feature from USE helps improve the classification results. Figure 5.8b depicts the behavior of DecAtt with the different text representations. With this model, there is a bigger gap between the representations, as compared to SiameseGRU and ESIM. The results show that Glove, both with or without USE, clearly outperforms both versions of FastText in combination with this model. Figure 5.7c illustrates the performance of ESIM with the different text representations. The results show that GloVe is highly competitive with FastText and Glove + USE. We also notice that ESIM seems more robust to the type of pre-trained embedding used, and is able to extract useful information from all text representations. On the other hand, for the smallest samples (Quora 5k) Glove remains the best performing text representation.

Figure 5.8 shows the results on AskUbuntu samples. The text representations show the same trend as on Quora dataset, with USE improving the baseline by a small margin. This is surprising, because we expected that FastText will be slightly more beneficial on this dataset considering the technical terms which might not be found in the pre-trained embeddings and would need to be computed as the sum of their n-grams. Generally, both Glove and FastText give a similar performance.

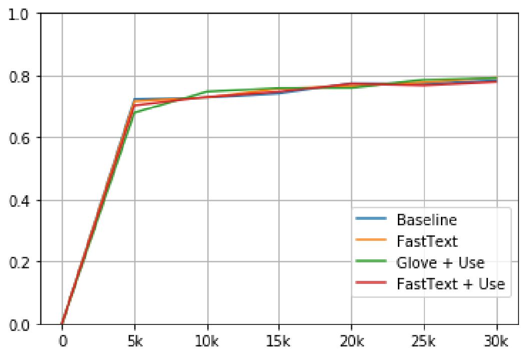
Based on these results we cannot conclude which of the pre-trained embeddings would be more suitable for the task of identifying duplicate questions. GloVe seems to have a more similar vocabulary to Quora and AskUbuntu based on Tables 5.14, 5.15, with the subword information from FastText not making much of a difference overall. On the other hand, the results show that using an additional feature from USE can bring improvements in the classification results. Better ways to incorporate the sentence embeddings exceed the scope of this thesis (e.g. use different distance metrics), but would constitute an interesting study area. Moreover, the experiment shows that the choice of pre-trained embeddings is model dependent, as the siamese network benefits from having embeddings for the all words, but for the two sentence interaction models there is no consistent result. Surprisingly, the pattern in performance looks more similar in SiameseGRU and ESIM, compared to DecAtt. We believe this is because both SiameseGRU and ESIM use recurrent neural networks (GRU, respectively BiLSTM), unlike DecAtt which mostly makes use of mathematical computations to build the question representation. Another important factor is that the SiameseGRU network updates the embedding matrix during training, fine-tuning the weights to the datasets, which likely



(a) SiameseGRU.



(b) DecAtt.



(c) ESIM.

Figure 5.7: Results of *Experiment 2: Text Representation* on Quora samples. The different text representations are highly competitive, with USE achieving a slightly improved performance.

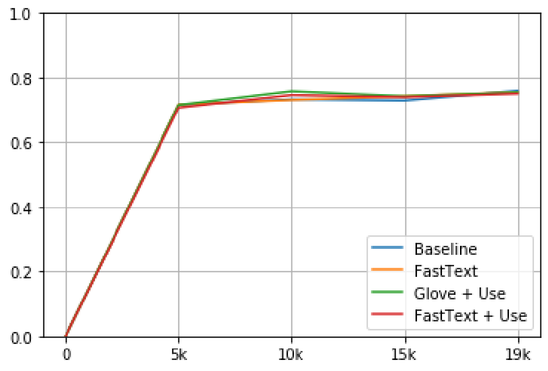
leads to a similar performance of all text representation techniques.

It would be interesting to experiment with more deep learning models in the same setting in order to clarify if one of the embeddings has a consistent performance. For now, we can state that using sentence embeddings from USE is definitely worth taking into consideration when dealing with the task of identifying duplicate questions from small data samples, while Glove and FastText give competitive results.

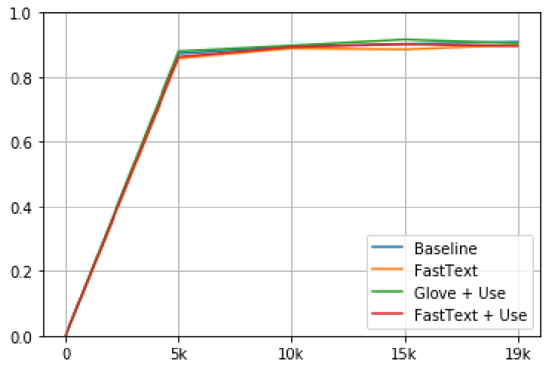
5.3.3 Effect of data augmentation

We focus on the three smallest samples from Quora and AskUbuntu, ranging from 5k to 15k pairs of questions. We apply the techniques discussed in Section 4.3, namely reverse questions, thesaurus augmentation, heuristic augmentation and noise embedding perturbation from Gaussian distribution.

Figure 5.9 illustrates the results of this experiment on Quora dataset. SiameseGRU baseline benefits from almost all proposed data augmentation techniques across the dif-



(a) SiameseGRU.

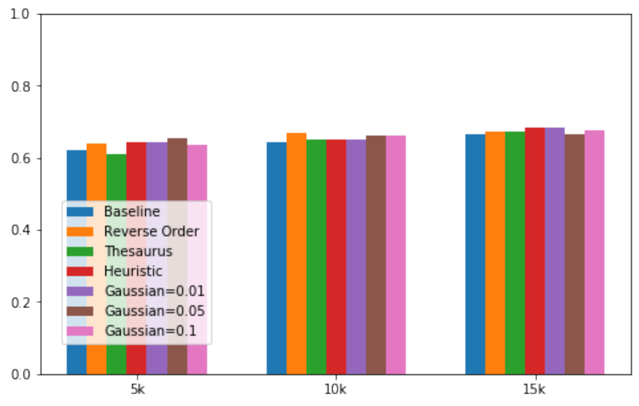


(b) DecAtt.

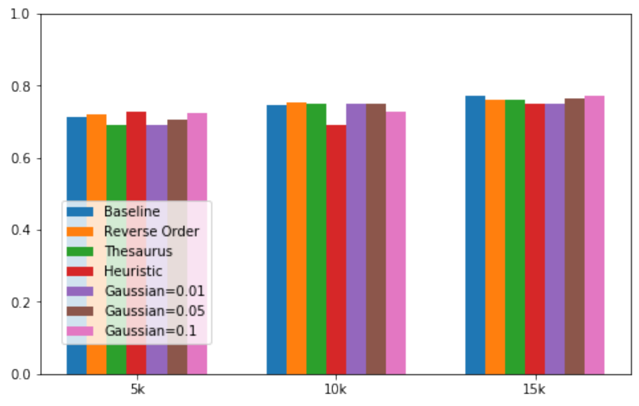
Figure 5.8: Results of *Experiment 2: Text Representation* on AskUbuntu samples. Similarly to Quora the only improvement is brought by USE.

ferent samples. DecAtt also benefits from the techniques for the smallest Quora sample, but this effect is alleviated as the sample size increases. ESIM displays the opposite behavior, with the baseline giving a good performance on the smallest sample size(only slightly exceeded by the reverse order technique) and being consistently outperformed by different augmentation techniques as the size increases.

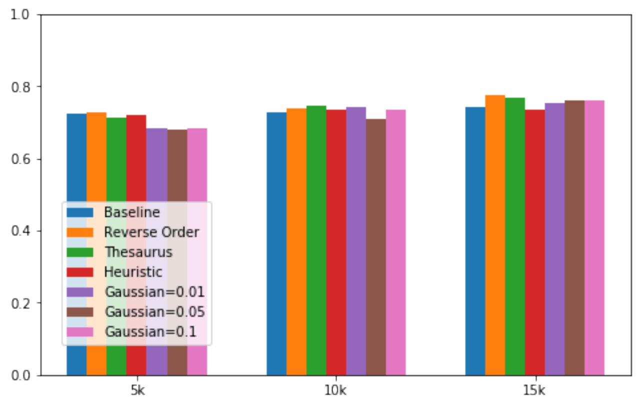
The results show that a simple data augmentation technique, such as reversing question order, can be effective in improving the performance of neural networks on general datasets like Quora. As all models include subtraction in the way they build the question representation, the vectors for a pair of questions and its reverse will be different because subtraction is not a symmetric operation. Thus, the models will have more data to learn from. Homma et al. also observe this effect in their experiments with data augmentation, but in their case they only try it on the siamese network [Homma et al., 2017]. On the other hand, Thesaurus-based augmentation becomes effective as the size of the sample increases. In the case of the smallest Quora sample (5k) it gives worse results than the baseline which shows the impact of unintentional noise from replacing



(a) SiameseGRU.



(b) DecAtt.



(c) ESIM.

Figure 5.9: Results of *Experiment 3: Data Augmentation* on Quora samples.

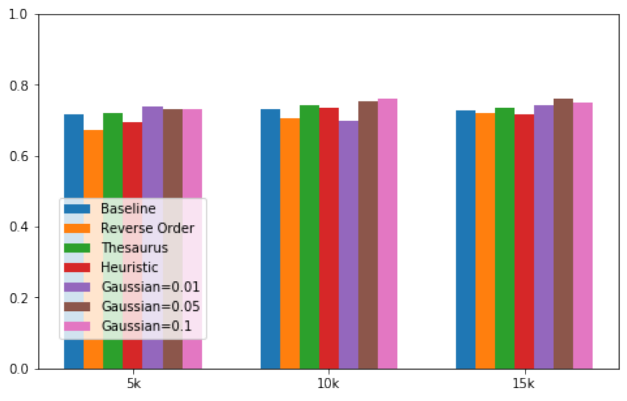
words with their synonyms (which are rarely exact) when data is scarce. In contrast, as the size of the sample increases (10k, 15k) the models become more robust. A different approach to this would be to replace the word with its synonym based on the closest word embedding, instead of using an external knowledge base like WordNet.

The results suggest that the augmentation method using a heuristic to create more data is the least effective. Overall, this technique creates up to 40% more data, slightly reducing the initial imbalance in the data, as the 5k, 10k and 15k samples now have 39.12%, 43.32%, respectively 38.05% duplicates instead of $36-37\%$. It benefits the smallest sample, but does not provide consistent results for the samples with more data. More data clearly helps the models give better performance on the 5k sample, but seems to add more noise to larger samples. This may be because this technique increases the mean lexical overlap of the samples and the models do not have the lexical knowledge to better understand these ambiguous pairs of questions. Glockner et. al show that sentences with very high overlap can "break" models like ESIM and DecAtt [Glockner et al., 2018]. This is also reflected in our results, with the samples augmented by this method performing worse for DecAtt and ESIM on Quora 10k and 15k samples.

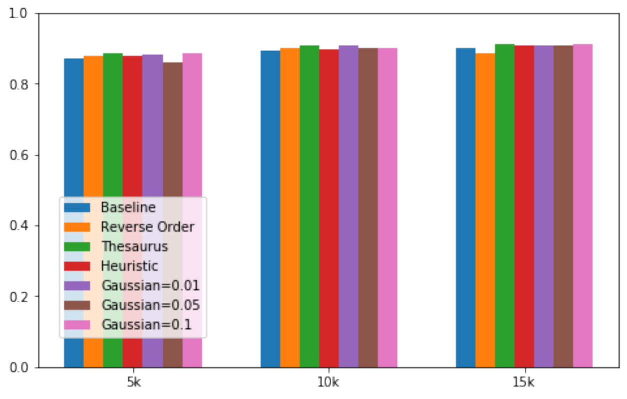
In terms of word embedding perturbation, by adding various amounts of noise to the embedding matrix we do not see major improvements on the baseline. Overall, the results indicate that this approach is not useful for DQD on datasets with high word overlap like Quora, considering that by applying small changes to the embedding matrix we might change the meaning of a word, and indirectly, the meaning of the sentence. This can occur especially if the pair of questions have a lot of words in common and by adding random noise to their embedding, the same word will have slightly different embeddings. However, we see an improvement on the SiameseGRU baseline, which is surprising considering that we expected this model to be more sensitive to noise. It still underperforms compared to DecAtt and ESIM, but it achieves up to 0.05 higher F1-score compared to the baseline. This indicates that with further fine-tuning the amount of noise to be added this technique can be useful for sentence encoding models.

Figure 5.11 illustrates the result of this experiment on AskUbuntu dataset. Overall, ESIM is not influenced much by the different techniques, achieving high F1-scores throughout the entire experiment. SiameseGRU is affected by some of the techniques, but the overall improvement is relatively small (at most 0.03 when applying $\{0.01\}$ Gaussian noise to AskUbuntu 5k and $\{0.05\}$ Gaussian noise to AskUbuntu 15k). We observe that reversing the order of questions is less consistent for this dataset (outperforming only the ESIM baseline on 5k and 10k samples), while the datasets augmented with questions that had their words replaced with synonyms slightly outperforms all the baselines. One possible explanation for why the models are more robust to Thesaurus augmentation in this setting, is that the questions on AskUbuntu are technical and technical words typically do not have multiple meanings (i.e. polysemy). This means that less words are replaced by synonyms because a lot of the terms refer to specific aspects of Ubuntu OS and do not have synonyms, but also because sentences contain on average approximatively 40 stop words, which also do not typically have synonyms.

Similarly to Quora results, the heuristic based method is not improving the results,



(a) SiameseGRU.



(b) ESIM.

Figure 5.10: Results of *Experiment 3: Data Augmentation* on AskUbuntu samples.

although ESIM is more robust to it than on Quora dataset. This is likely because, as Table 5.7 shows, most of the added questions have low overlap. Surprisingly, adding Gaussian noise to the embedding matrix is less disruptive for ESIM on this dataset, compared to Quora. The most notable effect of Gaussian noise can be observed on the SiameseGRU model, similarly to what we have seen on Quora samples, confirming that this type of noise can be beneficial for sentence encoding models. Overall, this method seems more robust for this dataset, probably because of the low word overlap. Adding {0.1} Gaussian noise renders the most consistent results, marginally improving both models on all samples.

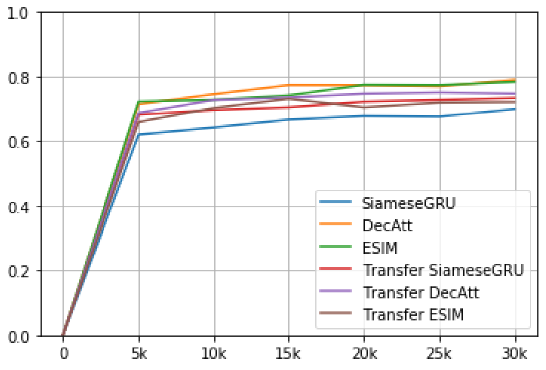
From this section we can conclude that a simple technique such as reversing the order of questions is slightly more beneficial, especially for models that build the question representation using asymmetrical operations (such as subtraction). Thesaurus based augmentation is also useful, but it mostly depends on how robust the model is to noise in the sentences and how general/technical the words in the dataset are. Heuristic augmentation is a good technique to test the lexical knowledge of the model, but not for creating new data for better classification results. Finally, word embedding perturbation, with varying amounts of Gaussian noise, is the promising on the sentence encoding model. The results also suggest that Gaussian noise is more robust when the pair of questions have a low word overlap. Generally, the improvements are small, which indicates that the models even more data for better performance.

5.3.4 Effect of transfer learning

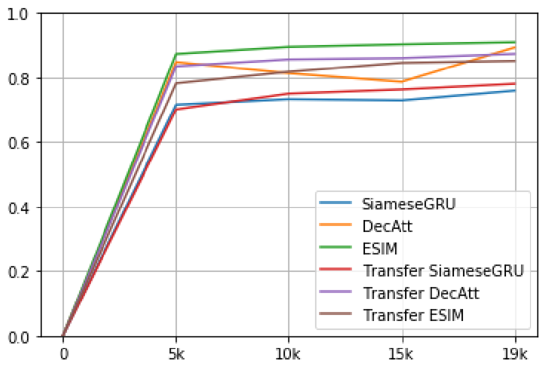
As explained in Section 4.4, we explore two strategies for transfer learning: entire model transfer and pre-training and fine-tuning a model. These strategies are tried in two different situations, specifically, knowledge transfer between similar datasets but different problems and knowledge transfer between the same problem and different datasets.

The results show that entire model transfer does not work well for the models that we have included in our framework. When we train the model on Quora Full and test it on AskUbuntu test set (no training data required in this scenario), all the models achieve very poor F1-score for duplicates: SiameseGRU identifies 0.09 duplicates, DecAtt identifies 0.02 duplicates and ESIM identifies 0.03 duplicates. This is because the two datasets have very different characteristics, even though they address the same problem. The most likely cause for this is that the models learn from Quora dataset that duplicate questions have high word overlap (see Section 5.1.1), which is not the case for AskUbuntu questions. On the other hand, when we transfer the models learned on the SNLI dataset to Quora, we notice that the results are better than before, with SiameseGRU identifying 0.50 duplicates, DecAtt identifying 0.36 duplicates and ESIM identifying 0.32 duplicates. This is because both SNLI and Quora have similar characteristics in terms of word overlap and sentence length. Overall, the results show that this method does not work well for identifying duplicate questions.

Figure 5.11a illustrates the effect of pre-training the models on SNLI dataset and fine-tuning on Quora dataset. We have experimented both with not freezing the weights of the model and freezing everything except the classification layers and we noticed that



(a) Pre-training on SNLI dataset and fine-tuning on Quora.



(b) Pre-training on Quora dataset and fine-tuning on AskUbuntu.

Figure 5.11: Results of *Experiment 5: Pre-training and fine-tuning.*

the most promising performance was achieved by allowing all weights to be updated. We also experiment with different learning rates and find that keeping the same value as when training the models (0.001) is more beneficial. The results show that SiameseGRU benefits the most from this technique, with the transfered model outperforming the baseline SiameseGRU on all samples. Quora 5k is 0.06 better in this case. On the other hand, ESIM and DecAtt baselines remain the best performing models overall. This shows that sentence encoding models are more robust across datasets and can effectively leverage the knowledge learned on a similar training task. Moreover, this also shows the clear advantage training on more data brings for siamese models. The lower results of the transfered DecAtt and ESIM is most likely because they would require more training data in order to substantially improve. This is probably because the interaction between sentences from SNLI dataset is slightly different than the interaction between duplicate questions (as explained in Section 3.2) and these models are more sensitive to this change.

Furthermore, we observe the same trend when we pre-train the model on Quora dataset and fine-tune on AskUbuntu samples. Figure 5.12b shows the results of this setting, where SiameseGRU baseline is outperformed by the transfered SiamereGRU, while ESIM baseline maintains its strong position. Surprisingly, if we fine-tune the model trained on Quora dataset the results of DecAtt are more stable than if that model was trained solely on AskUbutu samples. This is likely because the base model is stable, which helps the training in this case as the model does not need to learn everything from scratch.

From this section we can conclude that these models are not suited for entire model transfer, irrespectively of how similar the datasets they are trained on are. On the other hand, sentence encoding models definitely benefit from training the model on a different dataset and fine-tuning it on the target dataset. Sentence interaction models are more sensitive to the learned question representations and would probably require more data before fine-tuning on a target dataset can become effective.

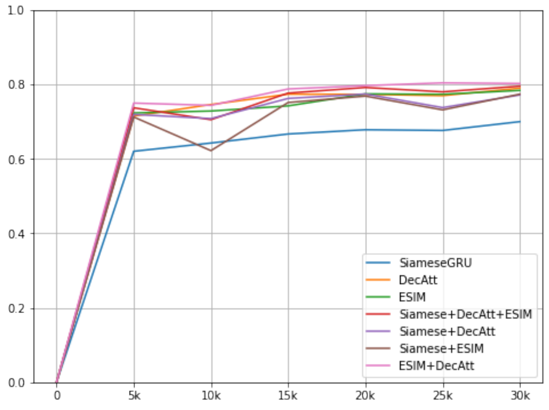
5.3.5 Effect of ensembling

In this section we present how ensembling can help with identifying duplicate questions. As explained in Section 4.5, we create ensembles by stacking the three models and by pairing them with each other. Hence, we have the following scenarios: 1) SiameseGRU, DecAtt, ESIM; 2) SiameseGRU, DecAtt; 3) SiameseGRU, ESIM; 4) ESIM, DecAtt.

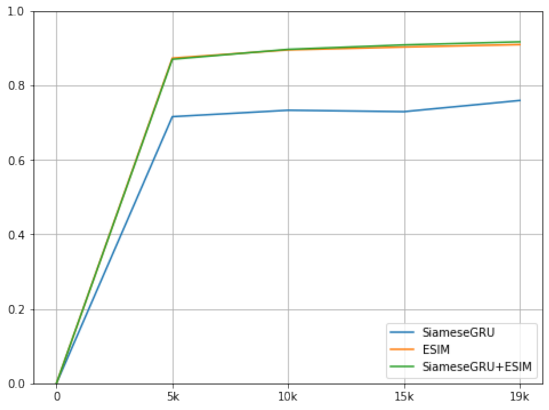
Figure 5.12a displays the effect of ensembling on Quora samples. The results clearly show ensembling is a very compelling technique, outperforming the baseline in almost all cases. The best ensemble is ESIM+DecAtt, which shows the predictive power that sentence interaction models have. In contrast, ESIM+SiameseGRU seems to be the least effective ensemble. Overall, the ensembles that contain the SiameseGRU model give a poorer performance compared to ESIM+DecAtt, most likely because the SiameseGRU is not able to capture the context and understand the interaction between the two questions. In fact, Figure 5.12a shows a clear pattern: all the ensembles where SiameseGRU is included have drops in F1-score on the same samples: 10k and 25k, The baseline (see Figure 5.6b) also shows SiameseGRU as the least performing model, so this result is not surprising. The highest F1-score score is achieved on the 25k and 30k samples: 0.8036, respectively 0.8019 F1-score.

Figure 5.12 depicts the results of ensembling on AskUbuntu dataset. As we have previously observed that DecAtt is highly volatile on this dataset, we could only ensemble SiameseGRU with ESIM and compare to the baselines. As we have seen that ESIM is very robust on this dataset, in this case combining it with SiameseGRU outperformed both baselines. Specifically, as the dataset size increases we observe a small increase in performance compared to ESIM baseline (up to 0.007 F1-score).

From this section we can conclude that ensembling different models is a powerful technique for better predicting which questions are duplicate and which are not. Ensembling overcomes the weaknesses of each individual model and has the potential to achieve better classification results.



(a) Quora.



(b) AskUbuntu.

Figure 5.12: Results of *Experiment 6: Ensembling*

5.3.6 Hybrid Solution

In this section we evaluate the performance of our hybrid solution. In Sections 5.3.1 to 5.3.5 we have explored and evaluated different strategies for text representation, data augmentation, transfer learning and ensembling. Based on the results, we create a hybrid solution from our best performing components and we evaluate it against the baseline. As our components performed slightly different on our datasets, each hybrid solution is tailored to the dataset.

The hybrid solution for AskUbuntu is obtained by ensembling ESIM and SiameseGRU, using GloVe and USE for text representation and augmenting the dataset either through Thesaurus augmentation or word embedding perturbation with Gaussian noise={0.1}. Hence, we test two solutions, where the data augmentation component differs. We have opted for this approach because both augmentation methods marginally improve the baselines. Based on the evaluation of the ensemble of models trained in this setting, neither of these solutions improves the ESIM baseline, while both are better than SiameseGRU baseline. This reflects the general behavior of ESIM and SiameseGRU that we have seen so far and shows that the small improvements that we see in Figure 5.12 is not significant. Overall, this indicates that further research is needed for DQD when datasets have long questions with low word overlap.

Figure 5.13 shows the performance of our best solution on Quora samples. For this experiment we have chosen GloVe and USE for text representation, reversing question order as data augmentation and ensembling ESIM and DecAtt. Transfer learning was not included because it did not have consistent results for all the models. The results show that the combination of these methods outperforms all baselines on all samples. The most important increase in performance we see on the smallest sample (Quora 5k), where F1-score improves by 0.04.

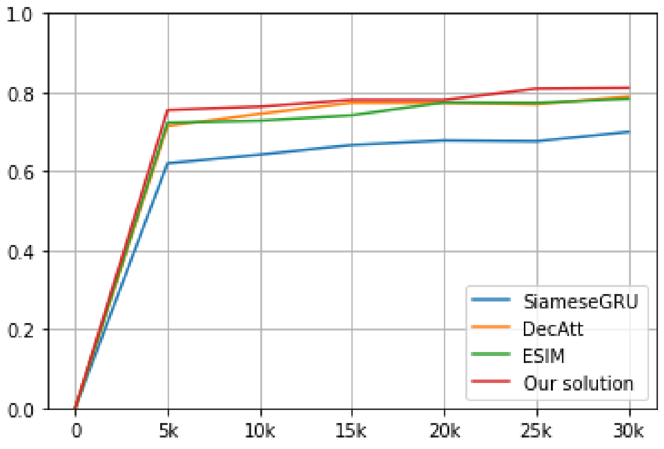


Figure 5.13: Best performing components applied to Quora dataset. The solution consists of using GloVe and USE for text representation, Reverse Order technique for data augmentation and ensembling ESIM and DecAtt. All the baselines are outperformed by our hybrid solution.

There are 621 questions common to all Quora samples that this solution labels incorrectly. The majority of them (429) are non-duplicates. One important characteristic of this group of questions is the high word overlap. On average these questions have an overlap of 64%, which is much higher than the overlap the samples have (which is about 52%). Table 5.16 shows an overview of the types of questions that are mistaken by our solution. Based on the classification introduced in Section 1.2 the main types of paraphrases can also be seen in this table: pair 1 is obtained through *morphology-based changes*, pairs 2 and 4 are obtained through *lexicon-based changes* and pair 3 is obtained through *semantic-based changes*. As for non-duplicates, pairs 6 and 7 differ only through on word, while questions 5 and 8 talk about the same subject, with a subtle difference. These types of questions where the differences between the pairs are small are still challenging for deep learning models to classify correctly. The next step in improving the classification of duplicate questions is to specifically tackle these types of questions and remains to be resolved by future work.

ID	Question 1	Question 2	Label
1	Are democracy inherently inefficient?	Are democracies inherently inefficient ?	Duplicate
2	What is singleton class in java?	What is a singleton class used for?	Duplicate
3	What is the best way to give speech?	How do I give a killer speech ?	Duplicate
4	What should I do to avoid sleeping in class?	How do I not sleep in a boring class?	Duplicate
5	What is your favorite type of toy dog?	What is your favorite dog and why?	Non-Duplicate
6	How can I improve my skills?	How do I improve my learning skills ?	Non-Duplicate
7	What is your opinion about showflipper.com?	What is opinion about Pippli.com?	Non-Duplicate
8	What is exactly there after life?	Is there life after life?	Non-Duplicate

Table 5.16: Example of questions from Quora samples mislabeled by the hybrid solution.

6 Conclusion and Future Work

This chapter summarizes the main ideas and results from this thesis. Our goal is to advance the study of duplicate question detection by identifying techniques to improve the performance of deep learning models when training data is scarce. We propose and evaluate four different components for achieving this goal: text representation, data augmentation, transfer learning and ensembling. The experiments are performed on two competitive datasets with very different characteristics.

Overall, the content of the questions does not influence much the final classification result. Aspects such a word overlap, sentence length and noise in the labels influence more the classification result. The results show that Glove and FastText pre-trained word embeddings are very competitive text representations. We demonstrated that an additional feature from the Universal Sentence Encoder can bring a small boost of performance to the deep learning models when datasets are small. In terms of data augmentation, we showed that reversing the order of questions as means of doubling the size of the original dataset can be helpful, especially if the models build the question representation using asymmetrical operations like subtraction. Creating new pairs of questions by replacing words with their synonyms is also helpful, but it depends on the datasets and how technical they are. Specifically, this method works better on technical datasets where words do not have polysemy. We proposed a new method for data augmentation, namely adding the pairs of questions which fit the criteria of having high overlap or that have low overlap and are duplicates. While this method did not perform well in terms of increasing F1-score, we believe this method can be used to test the lexical knowledge of deep learning models as questions with high word overlap or just one different word can pose a great challenge to deep learning models. We also showed that small amounts of Gaussian noise added to the embedding matrix makes sentence encoding models more robust. In addition, from the transfer learning experiment we can conclude that transferring knowledge between datasets that address different problems but have similar characteristics is better than very different datasets that address the same problem. At the same time, pre-training and fine-tuning a model highly depends on the model's ability to capture and transfer context and in this case it performed well only for the sentence encoding model. Finally, our results show that ensembling by stacking different deep learning models is a very compelling technique for achieving better classification results.

As a conclusion, we propose a hybrid solution which consists of using GloVe and Universal Sentence Encoder for text representation, reversing question order as data augmentation and ensembling two sentence interaction models. This solution is effective in outperforming all baselines in a setting with limited training sets on a general purpose dataset like Quora, where questions are short and the pairs share more words. On the other hand, how classification can be improved in datasets with longer sentences and fewer words in common remains to be studied by future work.

Identifying duplicate questions is a very challenging task that entails that deep learning models are able to understand semantics and syntax. In order to further improve the performance of deep learning models on small data samples, one challenge that remains to be addressed is that of questions with very high word overlap or that differ only through small details. For models it is hard to differentiate between these questions [Glockner et al., 2018], so it is important to either equip existing models with more lexical information or build models that address this issue. A possible way to achieve this is to identify words that destroy labels (i.e. salient words) and propose a weighting scheme that would put more emphasis on these words. Samanta and Mehta propose a method for ranking the importance of the words in context in order to generate adversarial examples, so this could be a starting point [Samanta and Mehta, 2017].

We have discussed the role of data augmentation in helping models generalize and reduce overfitting for small datasets. Creating new, diverse data is challenging, as our results show. Another method for data augmentation is back-translation [Sennrich et al., 2016]. Back-translation refers to creating synthetic training data by choosing a "pivot" language (for example German), and translating the pairs of questions to that language and then back to English. This approach is somehow similar to Thesaurus augmentation but it is likely to create the new questions closer to natural language (based on the quality of the system used for translation). A new technique for data augmentation for NLP problems is style-transfer [Fu et al., 2017]. This approach can be used to transfer certain aspects of a sentence (e.g. formality) without modifying its content [Prabhumoye et al., 2018]. For duplicate question detection it can create new data either by modifying certain attributes of a question and pairing it to the original or by applying two different styles to the same question and creating a new pair. In terms of word embedding perturbation, deep learning models could be trained by adding adversarial noise to the embedding matrix [Zhang and Yang, 2018]. In adversarial training [Goodfellow et al., 2014] the goal is to generate examples that are slightly different from the original input and would "fool" the classifier. Adversarial noise, as proposed by Zhang et al., would be added in the direction that would maximally increase the loss function. They discuss two other variants, Gaussian Adversarial Noise and Bernoulli Adversarial Noise that are promising on the datasets they use for evaluation. These methods have not been tried yet for duplicate question detection. Furthermore, You et al. propose the Adversarial Noise Layer as a means to improve the generalization of CNN for image classification [You et al., 2018]. They discuss the possibility of extending this approach to NLP, so it would be interesting to evaluate it on the problem of duplicate question detection.

As a final remark, the NLP community is leaning more and more towards the transfer learning paradigm, aiming to pre-train complex architectures that could be applied to many tasks [Howard and Ruder, 2018], [Radford et al., 2018]. Thus, finding a suitable pre-training task or an architecture better suited for transfering between pair-wise classification tasks, would be a step forward in solving the duplicate datection problem.

List of Acronyms

CNN	Convolutional Neural Network
DL	Deep Learning
DQD	Duplicate Question Detection
GRU	Gated Recurrent Unit
LSTM	Long Short Term Memory Network
MLP	Multilayer Perceptron
NLI	Natural Language Inference
NLP	Natural Language Processing
NN	Neural Network
PI	Paraphrase Identification
QA	Question-Answering (i.e. Answer Sentence Selection)
Q&A	Question-and-Answer
RNN	Recurrent Neural Network
TF-IDF	Term Frequency - Inverse Document
SNLI	Stanford Natural Language Inference

Bibliography

[neu, 2018] (2018). *Neuron*. Wikipedia. `https://en.wikipedia.org/wiki/Neuron`.

[Addair, 2017] Addair, T. (2017). Duplicate question pair detection with deep learning.

[Arora et al., 2016] Arora, S., Liang, Y., and Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bangal, 2009] Bangal, B. C. (2009). Automatic generation control of interconnected power systems using artificial neural network techniques.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Berger et al., 2000] Berger, A., Caruana, R., Cohn, D., Freitag, D., and Mittal, V. (2000). Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–199. ACM.

[Bilenko et al., 2003] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

[Bogdanova et al., 2015] Bogdanova, D., dos Santos, C., Barbosa, L., and Zadrozny, B. (2015). Detecting semantically equivalent questions in online user forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 123–131.

[Bojanowski et al., 2016] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

[Bowman et al., 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

[Bromley et al., 1994] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744.

[Cer et al., 2018] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175.*

[Chen et al., 2016] Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., and Inkpen, D. (2016). Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038.*

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

[Chollet, 2015] Chollet, F. (2015). keras. `https://github.com/fchollet/keras`.

[Chung et al., 2016] Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147.*

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555.*

[Cochinwala et al., 2001] Cochinwala, M., Kurien, V., Lalk, G., and Shasha, D. (2001). Efficient data reconciliation. *Information Sciences*, 137(1-4):1–15.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

[Conneau et al., 2017] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364.*

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

[Dolan and Brockett, 2005] Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005).*

[Draisbach and Naumann, 2010] Draisbach, U. and Naumann, F. (2010). Dude: The duplicate detection toolkit. In *Proceedings of the International Workshop on Quality in Databases (QDB)*, volume 100000, page 10000000.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

[Elmagarmid et al., 2007] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16.

[Finnoff et al., 1993] Finnoff, W., Hergert, F., and Zimmermann, H.-G. (1993). Extended regularization methods for nonconvergent model selection. In *Advances in Neural Information Processing Systems*, pages 228–235.

[Firat et al., 2016] Firat, O., Cho, K., and Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073.*

[Fu et al., 2017] Fu, Z., Tan, X., Peng, N., Zhao, D., and Yan, R. (2017). Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861.*

[Glockner et al., 2018] Glockner, M., Shwartz, V., and Goldberg, Y. (2018). Breaking nli systems with sentences that require simple lexical inferences. *arXiv preprint arXiv:1805.02266.*

[Goldberg, 2017] Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

[Gong et al., 2017] Gong, Y., Luo, H., and Zhang, J. (2017). Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348.*

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[Goodfellow et al., 2014] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572.*

[Gravano et al., 2003] Gravano, L., Ipeirotis, P. G., Koudas, N., and Srivastava, D. (2003). Text joins in an rdbms for web data integration. In *Proceedings of the 12th international conference on World Wide Web*, pages 90–101. ACM.

[Grave et al., 2018] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).*

[Graves et al., 2009] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

[Hardy et al., 1988] Hardy, G. H., Littlewood, J. E., and Pólya, G. (1988). *Inequalities.* Cambridge university press.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Homma et al., 2017] Homma, Y., Sy, S., and Yeh, C. (2017). Detecting duplicate questions with deep learning.

[Hoogeveen, 2018] Hoogeveen, D. (2018). *Real and misflagged duplicate question detection in community question-answering.* PhD thesis.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 328–339.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR*, volume 1, page 3.

[Jaro, 1980] Jaro, M. A. (1980). *UNIMATCH, a Record Linkage System: Users Manual.* Bureau of the Census.

[Jozefowicz et al., 2015] Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350.

[Kalchbrenner and Blunsom, 2013] Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

[Kim et al., 2018] Kim, S., Hong, J.-H., Kang, I., and Kwak, N. (2018). Semantic sentence matching with densely-connected recurrent and co-attentive information. *arXiv preprint arXiv:1805.11360.*

[Kinga and Adam, 2015] Kinga, D. and Adam, J. B. (2015). A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5.

[Konda et al., 2016] Konda, P., Das, S., Suganthan GC, P., Doan, A., Ardalan, A., Ballard, J. R., Li, H., Panahi, F., Zhang, H., Naughton, J., et al. (2016). Magellan: Toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208.

[Kozareva and Montoyo, 2006] Kozareva, Z. and Montoyo, A. (2006). Paraphrase identification on the basis of supervised machine learning techniques. In *Advances in natural language processing*, pages 524–533. Springer.

[Kumar et al., 2016] Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., and Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

[Lan and Xu, 2018] Lan, W. and Xu, W. (2018). Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. *arXiv preprint arXiv:1806.04330*.

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

[Luong et al., 2014] Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Miller, 1995] Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Olah, 2015] Olah, C. (2015). *Understanding LSTM Networks*. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[Papadakis et al., 2018] Papadakis, G., Tsekouras, L., Thanos, E., Giannakopoulos, G., Palpanas, T., and Koubarakis, M. (2018). The return of jedai: end-to-end entity resolution for structured and semi-structured data. *Proceedings of the VLDB Endowment*, 11(12):1950–1953.

[Parikh et al., 2016] Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Perone et al., 2018] Perone, C. S., Silveira, R., and Paula, T. S. (2018). Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259.*

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365.*

[Philips, 2000] Philips, L. (2000). The double metaphone search algorithm. *C/C++ users journal*, 18(6):38–43.

[Prabhumoye et al., 2018] Prabhumoye, S., Tsvetkov, Y., Black, A. W., and Salakhutdinov, R. (2018). Style transfer through multilingual and feedback-based back-translation. *arXiv preprint arXiv:1809.06284.*

[Qian, 1999] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

[Ravikumar and Cohen, 2004] Ravikumar, P. and Cohen, W. W. (2004). A hierarchical graphical model for record linkage. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 454–461. AUAI Press.

[Rodrigues et al., 2018] Rodrigues, J., Saedi, C., Branco, A., and Silva, J. (2018). Semantic equivalence detection: Are interrogatives harder than declaratives. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC2018). Accepted, to appear.*

[Rodrigues et al., 2017] Rodrigues, J. A., Saedi, C., Maraev, V., Silva, J., and Branco, A. (2017). Ways of asking and replying in duplicate question detection. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 262–270.

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

[Rücklé et al., 2018] Rücklé, A., Eger, S., Peyrard, M., and Gurevych, I. (2018). Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400.*

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

[Saedi et al., 2017] Saedi, C., Rodrigues, J., Silva, J., Maraev, V., et al. (2017). Learning profiles in duplicate question detection. In *Information Reuse and Integration (IRI), 2017 IEEE International Conference on*, pages 544–550. IEEE.

[Samanta and Mehta, 2017] Samanta, S. and Mehta, S. (2017). Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

[Sarawagi and Bhamidipaty, 2002] Sarawagi, S. and Bhamidipaty, A. (2002). Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278. ACM.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[Sennrich et al., 2016] Sennrich, R., Haddow, B., and Birch, A. (2016). Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

[Silva et al., 2018] Silva, J., Rodrigues, J., Maraev, V., Saedi, C., and Branco, A. (2018). A 20team's experiment and finding problems in its data preparation. In Branco, A., Calzolari, N., and Choukri, K., editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France. European Language Resources Association (ELRA).

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[Verykios and Moustakides, 2004] Verykios, V. S. and Moustakides, G. V. (2004). A generalized cost optimal decision model for record matching. In *Proceedings of the 2004 international workshop on Information quality in information systems*, pages 20–26. ACM.

[Vila et al., 2014] Vila, M., Martí, M. A., and Rodríguez, H. (2014). Is this a paraphrase? what kind? paraphrase boundaries and typology. *Open Journal of Modern Linguistics*, 4(01):205.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

[Wang and Jiang, 2016] Wang, S. and Jiang, J. (2016). A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

[Wang et al., 2017] Wang, Z., Hamza, W., and Florian, R. (2017). Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

[Wang et al., 2016] Wang, Z., Mi, H., and Ittycheriah, A. (2016). Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

[Winkler, 2002] Winkler, W. E. (2002). Methods for record linkage and bayesian networks. Technical report, Technical report, Statistical Research Division, US Census Bureau, Washington, DC.

[Yang et al., 2015] Yang, Y., Yih, W.-t., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

[Yin et al., 2015] Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2015). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

[You et al., 2018] You, Z., Ye, J., Li, K., and Wang, P. (2018). Adversarial noise layer: Regularize neural network by adding noise. *arXiv preprint arXiv:1805.08000*.

[Yu et al., 2018] Yu, J., Qiu, M., Jiang, J., Huang, J., Song, S., Chu, W., and Chen, H. (2018). Modelling domain relationships for transfer learning on retrieval-based question answering systems in e-commerce. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 682–690. ACM.

[Zhang and Yang, 2018] Zhang, D. and Yang, Z. (2018). Word embedding perturbation for sentence classification. *arXiv preprint arXiv:1804.08166*.

[Zhang and LeCun, 2015] Zhang, X. and LeCun, Y. (2015). Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.