UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
FACULDADE DE LETRAS
FACULDADE DE MEDICINA
FACULDADE DE PSICOLOGIA

Modelling Semantic Relations with Distributional Semantics and Deep Learning: Question Answering, Entailment Recognition and Paraphrase Detection

Vladislav Maraev

Thesis supervised by
Prof. António Branco, PhD
co-supervised by
Prof. Carlos Lourenço, PhD

MASTER IN COGNITIVE SCIENCE

2017

Resumo

Nesta dissertação apresenta-se uma abordagem à tarefa de modelar relações semânticas entre dois textos com base em modelos de semântica distribucional e em aprendizagem profunda. O presente trabalho tira partido de várias disciplinas da ciência cognitiva, com especial relevo para a computação, a linguística e a inteligência artificial, e com fortes influência da neurociência e da psicologia cognitiva.

Os modelos de semântica distribucional (também conhecidos como ”word embeddings”) são usados para representar o significado das palavras. As representações semânticas das palavras podem ainda ser combinadas para obter o significado de um excerto de um texto recorrendo ao uso da aprendizagem profunda, isto é, com o apoio das redes neurais de convolução.

Esta abordagen é utilizada para replicar a experiência realizada por Bogdanova et al. (2015) na tarefa de deteção de perguntas que podem ser respondidas as mesmas respostas tal como estas foram respondidas em fóruns on-line. Os resultados do desempenho obtidos pelas experiências apresentadas nesta dissertação são equivalentes ou melhores que os resultados obtidos no trabalho de referência mencionado acima.

Apresentao também um estudo sobre o impacto do pré-processamento apropriado do texto, tendo em conta os resultados que podem ser obtidos pelas abordagens adotadas no trabalho de referência supramencionado. Este estudo é levado a cabo removendo-se certas pistas que podem levar o sistema, indevidamente, a detetar perguntas equivalentes. Essa remoção das pistas leva a uma diminuição significativa no desempenho do sistema desenvolvido no trabalho de referência.

Nesta dissertação é ainda apresentado um estudo sobre o impacto que os word embeddings treinados previamente têm na tarefa de detetar perguntas semanticamente equivalentes. Substituindo-se, aleatoriamente, word embeddings previamente treinados por outros melhora-se o desempenho do sistema.

Além disso, o modelo foi utilizado na tarefa de reconhecimento de implicações para Português, onde mostrou uma taxa de acerto similar à da baseline.

Este trabalho também reporta os resultados da aplicação da abordagem adotada numa competição para a deteção de paráfrases em Russo. A configuração final apresenta duas melhorias: usa character embeddings em vez de word embeddings e usa vários filtros de convolução. Esta configuração foi testado na execução padrão da Tarefa 2 da competição relevante, e mostrou resultados competitivos.

Abstract

This dissertation presents an approach to the task of modelling semantic relations between two texts, which is based on distributional semantic models and deep learning. The present work takes advantage of various disciplines of cognitive science, mainly computation, linguistics and artificial intelligence, with strong influences from neuroscience and cognitive psychology.

Distributional semantic models (also known as word embeddings) are used to represent the meaning of words. Word semantic representations can be further combined towards obtaining the meaning of a larger chunk of a text using a deep learning approach, namely with the support of convolutional neural networks.

These approaches are used to replicate the experiment carried out, by Bogdanova et al. (2015), for the task of detecting questions that can be answered by exactly the same answer in online user forums. Performance results obtained by my experiments are comparable or better than the ones reported in that referenced work.

I present also a study on the impact of appropriate text preprocessing with respect to the results that can be obtained by the approaches adopted in that referenced work. Removing certain clues that can unduly help the system to detect equivalent questions leads to a significant decrease in system's performance supported by that referenced work.

I also present a study of the impact that pre-trained word embeddings have in the task of detecting the semantically equivalent questions. Replacing pre-trained word embeddings by randomly initialised ones improves the performance of the system.

Additionally, the model was applied to the task of entailment recognition for Portuguese and showed an accuracy on a level with the baseline.

This dissertation also reports on the results of an experimental study on the application of the adopted approach to the shared task of sentence paraphrase detection in Russian. The final set up contained two improvements: it uses several convolutional filters and it uses character embeddings instead of word embeddings. It was tested in Task 2 standard run of the relevant shared task and it showed competitive results.

Acknowledgements

I am grateful to my supervisor António Branco for his extraordinary support, for helping me with choosing the right directions all the way through my work on this dissertation, and for showing me how the research should be carried out in NLX-Group.

I also would like to thank my co-supervisor Carlos Lourenço for his help on early stages of my work and for his useful comments regarding the sensitive part of my dissertation.

Working at NLX-Group has always been fun and a pleasure. I would like to thank everyone in the group for their help, for sharing their experience and for keeping my mood up. Especially I would like to thank João Rodrigues for spotting the clues in the StackExchange datasets and João Silva for his friendly guidance and for his significant help with organising my code.

I think that the acknowledgement page is a right place to mention great tools that helped me a lot and made my procrastination more productive (and indeed interesting), namely Emacs, Org mode and LaTeX.

Lastly, my thanks go to my family and friends. Either in St. Petersburg or in Lisbon my work would be impossible without your support.

*

The work was partially supported by the Portuguese Government's P2020 program under the grant 08/SI/2015/3279: ASSET-Intelligent Assistance for Everyone Everywhere, and by the EC's FP7 program under the grant number 610516: QTLeap-Quality Translation by Deep Language Engineering Approaches.

Table of contents

Chapter 1

Introduction

Question Answering (QA) has been a long-standing goal in Artificial Intelligence. To pose the question to a knowledge base we should do our best to provide a semantic representation of a question. Therefore, QA is also a good task to assess natural language semantic representation theories.

There are different approaches to natural language semantics that can either be symbolic (e.g. truth-value semantics), or distributional (e.g. vector space semantics).

Many areas of language technology have recently explored deep learning and distributional semantic representation. In the present dissertation I am resorting to distributional semantic models and deep learning approaches to a question answering task.

1.1 Motivation

Document and information retrieval systems, such as GOOGLE, BING and many others, do a really good job by accepting a user query and retrieving a set of relevant documents and webpages. They lack, however, the ability to answer a query in a more natural and specific form, i.e. a form in which the question may be asked by a user in a natural language, and not just as a sequence of keywords.

Over the years, a great variety of question answering systems have been created. Some systems are intended to retrieve factoid answers in very specific domains, such as one of the first QA systems LUNAR (Woods and Kaplan, 1977) that answered questions about the geological analysis of rocks returned by the Apollo 11 moon missions. Other systems are more open-domain oriented such as IBM WATSON (Ferrucci, 2012) that could compete with champions at the game of Jeopardy!™.

There are dozens of textual question answering systems described in the literature. Following (Mitkov, 2005, Chapter 31) two basic types of question answering systems can be distinguished: systems that try to answer a question by using a structured information contained in a database, and systems that try to answer a question by analysing plain-text information. Of course, plain-text based systems can be enriched with structured data constituting a hybrid system aimed to provide more accurate reply.

The topic of question answering, along with many other areas of Artificial Intelligence (AI) and Natural Language Processing (NLP), recently received a huge research boost due to two main reasons: availability of massive amounts of data from the web and increasing growth of computational power, including by the use of Graphics Processing Units (GPUs). The area of question answering was also supported by the emergence of Web 2.0-based online QA communities, that enhance online information seeking by allowing users to exchange their knowledge in the form of asking and answering questions.

This enhanced the success of AI and a broad category of machine learning systems, that can learn to carry out different tasks just by being exposed to examples of the relevant raw input data and the corresponding output. Machine learning approaches play an important role in on the fly adaptation of systems and provides an option for question answering capabilities to be learned from the data itself.

My work seeks to apply models that are based on machine learning approaches: distributional semantic models (DSMs) and convolutional neural networks (CNNs) to the question answering task. DSM is a very effective approach to word meaning representation and CNN is a type of 'deep' neural networks, that achieved huge success, especially in the computer-vision area.

1.2 Contributions

My masters research work is driven by the goal of replicating the experiments and the excellent results obtained by Bogdanova et al. (2015) and by reusing them in further experiments. The authors of this paper applied CNNs and DSMs to detect semantically equivalent questions in online user forums, such as AskUbuntu and META StackExchange. Bogdanova et al. (2015) defined questions as semantically equivalent if they can be adequately answered by the exact same answer. In their study they have a goal to predict if two questions are semantically equivalent. A major part of my work is a replication of this study, that includes preparation of dataset, building DSM for word meaning representation and CNN for representing a similarity between pairs of questions.

My replication used similar dataset (but not exactly the same) and reaches very similar accuracy score for this task (results of Bogdanova et al. (2015) are shown in brackets): 94.1% (92.9%) for AskUbuntu dataset and 94.2% (92.7%) for META StackExchange dataset.

Additionally, in the present work I discuss that in the study undertaken by Bogdanova et al. (2015) there, presumably, exists a drawback regarding text preprocessing, which did not ensure the removal of certain clues that can indicate that a given question is a duplicate of other question. Removal of these clues significantly drops the performance of the system (to less 20.8 p.p. for the AskUbuntu dataset, for instance).

Another contribution of the dissertation, in the application of similar models to other tasks, is the textual entailment recognition for Portuguese and paraphrase detection for Russian.

In the Portuguese task, I used the ASSIN evaluation set (Fonseca et al., 2016) for the task of classifying the presence of textual entailment. The model that I had developed for the QA task showed easy adaptation for this new domain and task and it obtained a score of 69.1% in terms of classification accuracy.

In the Russian task, I used the ParaPhraser dataset (Pronoza et al., 2015) for the task of classifying the pair of sentences as paraphrases (precise or near paraphrases) or non-paraphrases. I participated in the Russian paraphrase detection shared task which was co-located with AINL 2016 and my solution obtained the fifth place with 72.74% accuracy (difference to the best team is 3.12%).

1.3 Impact in cognitive science

The present work connects different disciplines of cognitive science, mainly artificial intelligence, computation and linguistics, with a strong inspiration from neuroscience and cognitive psychology.

Regarding *neuroscience* my work takes advantages of using artificial neural networks (ANNs), that are inspired by biological neural networks, and it derives important features from them. The distributional semantic models (DSMs), widely applied in my work, are also supported by neuroscience. For further information about the role of neuroscience please refer to Section 3.3.2, regarding ANNs, and to Section 3.2.3 regarding DSMs.

Regarding *cognitive psychology* distributional semantic models are widely used to model and simulate various psychological phenomena. Please refer to Section 3.2.3 for specific examples.

1.4 Outline of the thesis

A brief overview of the thesis structure, chapter by chapter, is provided below.

Chapter 2: Task Overview This chapter provides an overview of the approaches to the question answering task and the review of the paper, by Bogdanova et al. (2015), to be replicated.

Chapter 3: Background This chapter introduces the fundamental concepts and methods used throughout this thesis. I start with describing different approaches to the task of question answering. Then, I provide an overview of distributional semantic models. Finally, I introduce the concept of deep learning and offer a brief description of convolutional neural networks.

Chapter 4: Implementation and evaluation This chapter describes the tools that were implemented to tackle the task defined in Chapter 2. I start by explaining how convolutional neural networks work in further detail. Then, I review the structure of corpora and continue with a description of how to handle the data and the implementation of the proposed architecture. This chapter concludes with a presentation of the results obtained.

Chapter 5: Portuguese Entailment Recognition Task This chapter describes the application of the proposed architecture to the Portuguese entailment detection task.

Chapter 6: Russian Paraphrase Detection Task This chapter describes the application of the proposed architecture to Russian paraphrase detection task.

Chapter 7: Conclusions The final chapter concludes this dissertation with a summary of the research contributions, followed by an outline of possible future work.

Appendix A: User guide The appendix consists of the User Guide describing how to download and run the code in order to replicate the present work.

Chapter 2

Task Overview

This chapter defines the major goal of the dissertation, namely the replication of the experiment undertaken by Bogdanova et al. (2015).

In Section 2.1, the goal of the task of Question Answering (QA) will be detailed together with making explicit the differences with respect to other tasks and with narrowing the scope of the task to domain-specific community question answering systems. Section 2.2 provides a definition of the task addressed by Bogdanova et al. (2015), namely the detection of duplicate questions that will be replicated in the dissertation.

2.1 Question Answering

2.1.1 Differences from Information Retrieval and Information Extraction

The main feature of a Question Answering (QA) system is the ability to identify or return answers to questions; both, a question and an answer should be in plain natural language. Firstly, it is important to make a distinction between Question Answering, Information Retrieval (IR) and Information Extraction (IE).

The goal of an Information Retrieval (IR) system is to retrieve relevant documents by matching a query against the collection of documents. For example, for a query 'acts that Barack Obama signed', an IR system should return documents containing the description of the acts signed by Barack Obama.

Information Extraction (IE) systems extract information from a set of documents according to a target template, that is predefined. For example, an IE system can extract the year, the authors and other features for a certain act.

Question Answering systems return, in turn, an answer in natural language to an input question also in natural language. For example, an answer such as ‘He signed it on October 28, 2009’ can be returned to the input question ‘When did Barack Obama signed Matthew Shepard Act’.

It might appear that for setting up a QA system, we just would need to combine IR and IE techniques, but that would not be viable because it would require extraction rules for all possible domains, and would restrict the types of questions that may be asked to the forms of information modelled by extraction templates (Mitkov, 2005, Chapter 31).

2.1.2 Open-domain vs Domain-specific QA

Question Answering technology can be very broad: it can help users ask simple factual question, e.g. about nature, politics, etc. On the other hand questions can be very specific: users may ask about the functioning of a specific device or a program, they can enter a query about stock market, or ask about very specific health or forensic information, etc.

Open-domain QA systems discover answers in large document collections, and use natural language processing tools to provide the most relevant and accurate answer. Examples of open-domain QA system are XisQuê, developed in our NLX–Natural Language and Speech Group[1], or IBM WATSON system, that was “taught” to play Jeopardy!™. Example 2.1.1 shows a correct question-answer pair from Jeopardy!™ quiz, however IBM WATSON gave a wrong reply “Toronto” (Markoff, 2011).

Example 2.1.1 (*Open-domain QA*)

Question:	U.S. city; its largest airport was named for a World War II hero; its second largest, for a World War II battle.
Answer:	Chicago.

In *domain-specific* (or canned) QA systems, a new input question is matched against a set of predetermined pairs of questions and answers. The domain is restricted and, if a question is congruent with the domain, this type of systems have more chances to perform better than open-domain systems. Example 2.1.2 shows a question-answer pair taken from WIKIPEDIA FAQ[2]; a domain-specific system can receive a question, decide whether its meaning is close to a meaning of a question

[1]XisQuê is described in Branco et al. (2008), and it is available at http://xisque.di.fc.ul.pt/en/
[2]https://en.wikipedia.org/wiki/Wikipedia:FAQ/Editing

stored in frequently asked questions database (FAQ DB) and extract an answer from the database.

Example 2.1.2 (*Domain-specific QA*)

Question:	How do I insert a new line?
Answer:	Normally, Wikipedia doesn't start a new line when you press the Enter key. If you press the Enter key twice, Wikipedia will start a new paragraph. To force a single new line (for instance, when you want to insert a poem) insert the HTML element `<br />` after the line.

2.1.3 Communities for Question Answering

The rapid development of the Internet and Web 2.0, that empowered the Internet with social interactivity, allowed the emergence of many online social communities that bring people together and help them to exchange knowledge.

Online Q&A communities are a subset of asynchronous online communities that facilitate knowledge sharing in the form of asking and answering questions. Typically, answers from a Q&A community are available for all users and indexed by search engines. Membership in Q&A communities is also available for all users, but not all users may be able to post questions and answer them as this may be restricted by some policies of a specific Q&A community. For example, StackExchange communities resort to user reputation[3] to control users' privileges on asking and answering questions, commenting them, voting for favourable answers, etc. Voting system and reputation helps Q&A communities to self-regulate, which brings more power to more relevant and expert answers.

These characteristics have resulted in the rapid growth of online Q&A communities during the last decade. For example, Yahoo! Answers, one of the largest English-language online Q&A communities, had 200 million users by December 2009, and over one billion questions-and-answers by October 2009[4]. Jin et al. (2013) made an online survey among Yahoo! Answers China community and claim that users' intention to continue answering questions (which is also an intention to share the knowledge) was directly influenced by two factors: satisfaction and knowledge self-efficacy.

[3]http://meta.stackexchange.com/questions/7237/how-does-reputation-work

[4]http://yanswersblog.com/index.php/archives/2009/12/14/yahoo-answers-hits-200-million-visitors-worldwide

Some Q&A communities are domain-specific, such as AskUbuntu, which is specialised on questions about the Ubuntu operating system. AskUbuntu is a part of Stack Exchange Network and in September 2016 it had about 376,000 registered users[5] and more than 240,000 questions[6]. In such domain-specific communities users' intention to keep answering questions can grow even further, with an intention of a user to become an active member of a certain community (i.e. Ubuntu users community), and not only a specific Q&A community (i.e. AskUbuntu).

This section has provided a background information for defining a task we are concerned with in this dissertation. Further information on background and state-of-the-art of question answering is given in Section 3.1.

2.2 Task Definition

2.2.1 Semantically Equivalent Questions

The (Bogdanova et al., 2015) is the study I seek to replicate, and this paper tackles an important characteristic of Q&A communities that wasn't mentioned in Section 2.1.3 above, which is a *duplication policy.* Typically in online Q&A communities nearly exact duplicates and copy-and-paste questions are quickly detected and removed from the website. However, some duplicates are kept and the main reason for that, as it is stated by AskUbuntu, is the following:

> *There are many ways to ask the same question, and users might not be able to find the answer if they're asking it a different way.*

Accordingly, Bogdanova et al. (2015) adopt the following definition for semantically equivalent questions:

Definition 2.2.1

> *Two questions are semantically equivalent if they can be adequately answered by the exact same answer.*

Examples 2.2.1 and 2.2.2 below are two questions taken from AskUbuntu community that were marked as duplicates. They state a problem in two different ways, however they can be answered by one answer (short version given):

[5] https://stackexchange.com/leagues/31/alltime/askubuntu/
[6] https://askubuntu.com/questions/

> *.exe files are not binary-compatible with Ubuntu. There are, however, compatibility layers for Linux, such as Wine, that are capable of running .exe*

When one steps into the domain of detecting semantically equivalent questions, one faces to basic problems, namely (1) the same question may be formulated in different ways; and (2) two questions may be asked about two different things but look for the same solution.

Example 2.2.1 (*Question*)

> Title: How can I install Windows software or games?
>
> Body: Can .exe and .msi files (Windows software) be installed in Ubuntu?

Example 2.2.2 (*Question, marked as a duplicate*)

> Title: I can't download anything and I can't watch videos [duplicate]
>
> Body: Two days ago I tried to download skype and it says an error occurred it says `end of central directory signature not found Either this file is not a zipfile, or it constitutes one disk of a multi-part archive. In the latter case the central directory and zipfile comment will be found on the last disk(s) of this archive. zipinfo: cannot find zipfile directory in one of /home/maria/Downloads/SkypeSetup-aoc-jd.exe or /home/maria/Downloads/SkypeSetup-aoc-jd.exe.zip, and cannot find /home/maria/Downloads/SkypeSetup-aoc-jd.exe.ZIP` <...>this happens whenever I try to download anything like games and also i can't watch videoss it's looking for plug ins but it doesn't find them i hate this

Bogdanova et al. (2015) mention several other tasks related to identifying semantically equivalent questions, namely (1) near-duplicate detection; (2) paraphrase identification; and (3) textual semantic similarity estimation. The main difference is these tasks are not concerned with searching for a relevant solution to a question, and are just looking for an alternative formulation of an utterance.

2.2.2 Architecture and Corpora

The architecture used by Bogdanova et al. (2015) combines two approaches in order to detect semantically equivalent questions: (1) *distributional semantic model* (DSM) for representing a meaning of each word in a question; and (2) *convolutional neural* *network* (CNN) for combining word meaning representations into a question representation. DSM ensures a vectorial representation of words, and words are combined by CNN into a vectorial representation of questions.

The architecture works with pairs of questions, and the resulting score, which is computed by similarity measure between vectors, supports a decision whether two questions are duplicates or not.

A main characteristic of the architecture is its machine learning capability that makes the QA system able to learn a better representations of a question from each new portion of labelled data.

Bogdanova et al. (2015) use two data dumps from two Q&A communities: (1) AskUbuntu forum that was briefly described in the previous section; and (2) META StackExchange community that is used to discuss about the StackExchange community itself. Bogdanova et al. (2015) prepare an input to their system in a form that contains a pair of two questions and a label showing if they are duplicates.

More detailed description of the architecture is provided in the next chapters: Chapter 3 will dwell on the notion of DSM and CNN, and in Chapter 4 I will give a detailed description of a CNN architecture that was replicated.

2.2.3 Results to be replicated

Firstly, the resulting model obtained by Bogdanova et al. (2015) outperforms SVM (support vector machine) baseline with a significant margin: accuracy 92.9% for CNN vs 82.4% for SVM (for AskUbuntu dump). And, this margin was even more significant on the limited training set.

Secondly, these authors show that their architecture is easily adaptable to different domains. They use data from META StackExchange community to show that with their model they are getting similar accuracy (92.68%) as the one obtained for the AskUbuntu community.

A major goal of my work is to replicate these results. Table 2.1 shows the results along with the characteristics of the corpora used by Bogdanova et al. (2015). It is expected that my work obtains comparable results within comparable settings.

Community	*Corpus date*	*Millions of tokens*	*Test Accuracy*
AskUbuntu	May 2014	121	92.90%
Meta	May 2014	19	92.68%

Table 2.1 Main results and corpora

Chapter 3

Background

This chapter introduces the most important concepts that will be used throughout the dissertation.

Section 3.1 reviews existing approaches to the task of question answering. In Section 3.2, I introduce the notion of *distributional semantics*, which is used to obtain a representation of the meaning of a word from the contexts in which it is used. The final Section 3.3 describes the *deep learning* computational techniques that are used to combine vectorial distributional representations of words into larger entities, such as phrases, sentences or paragraphs.

3.1 Approaches to Community Question Answering

We recall the definition of *semantic equivalence* of two questions:

Definition 2.2.1

Two questions are semantically equivalent if they can be adequately answered by the exact same answer.

There are different tasks that are aimed at detecting semantically equivalent utterances: duplicate and near-duplicate detection, paraphrase identification and textual semantic similarity prediction. Hereby we present state-of-the-art results for each of the tasks bearing in mind the difference with our task that was indicated in previous chapter: we are searching for a relevant solution to a question, and are not just looking for an alternative formulation of an utterance.

3.1.1 Duplicate and near-duplicate detection

The task of duplicate and near-duplicate detection is very important for search engines that deal with a list of retrieved documents that needs to be narrowed down to a list of unique documents in order to improve search results. Documents that have small dissimilarities are identified as *near-duplicates*. Approaches to near-duplicate detection can be broadly classified into *syntactic*, *URL based* and *semantic* techniques (Alsulami et al., 2012).

A state-of-the-art technique that is used for QA task was introduced by Wu et al. (2011). It is based on Jaccard coefficient, which is used to measure similarities between the two sets. They use various techniques to capture similarities between different parts of Q&A threads and, then, they use predefined threshold to determine whether threads are near-duplicates or not. Wu et al. (2011) approach was also used in the paper that we replicate (Bogdanova et al., 2015) as a baseline.

Semantically equivalent questions can have no word overlap, so duplicate detection task can not entirely address this issue.

3.1.2 Paraphrase identification

Two sentences with the same meaning are defined as paraphrases. A data set that is commonly used to support research on this task is Microsoft Research Paraphrase Corpus (Dolan et al., 2004). There are different state-of-the-art techniques for paraphrase identification, such as combination of machine translation solutions (Madnani et al., 2012), deep learning techniques (Cheng and Kartsaklis, 2015; He et al., 2015) and tensor factorization (Ji and Eisenstein, 2013).

3.1.3 Semantic Textual similarity

Semantic Textual Similarity (STS) measures the degree of semantic equivalence in the underlying semantics of paired snippets of texts. This task is very important for research in sentence-level semantics and there is a SemEval Shared Task dedicated to this issue that is held annually since 2012. Participating systems are asked to return a continuous valued similarity score on a scale from 0 to 5, with 0 indicating that the semantics of the sentences are completely independent and 5 signifying semantic equivalence. Performance is assessed by computing the Pearson correlation between machine assigned semantic similarity scores and human judgments (Agirre et al., 2016). Part of the test set uses data from Q&A online forums for making judgments about semantic similarity. The best system of SemEval-2015 uses word alignments and distributional semantic vector composition (Sultan et al., 2015).

3.2 Distributional Semantics

3.2.1 Forms of Meaning Representation

In cognitive science there has been a long debate regarding how semantic knowledge is organized and used in human language understanding and production. Known accounts of semantic representation can be grouped into three broad families, namely semantic networks, feature-based models and semantic spaces.

Semantic networks (Collins and Quillian, 1969) reproduce concepts as nodes in a graph whose edges denote semantic relationships between the concepts. For example, relation of hyponymy between the concept *salmon* and the concept *fish* can be represented as an edge of a graph that will connect two nodes that denote aforementioned concepts with indicating the relation between them. In semantic networks word meanings can be obtained by collecting relations from the graph. Semantic networks are representations that abstract away from real-world usage as they are constructed manually by human modelers.

As an alternative to the network approach, the meaning of a word can be described in terms of *feature lists* (Smith et al., 1974), that can be obtained by polling native speakers about what features they consider as relevant for the meaning of a word. This approach is limited due to the size of vocabulary and also the number and quality of attributes are highly dependent on time devoted to each word.

Semantic spaces, the third family of semantic representations is based on the assumption that word meanings are determined by linguistic environment (Wittgenstein, 1953). Words with similar meanings occur in similar contexts and one can talk about the common semantic value of expressions as word co-occurrence among them (Harris, 1954).

Distributional semantic models (DSMs) belong to the latter family of meaning representations and are typically implemented through vector space models, where the semantic representation for a word is a vector in a high-dimensional space. The dimensions stand for context items (for example, co-occurring words), and the coordinates depend on the co-occurrence counts or probabilities. Distributional models can be learned from a corpus in unsupervised fashion. Similarity between the semantic representations of words is usually computed using cosine similarity between the respective vectors.

3.2.2 Distributional Semantic Models Explained

Distributional semantic models are motivated by so-called *distributional hypothesis.* (Harris, 1954, page 156) stated the core assumption of it:

> *The fact that, for example, not every adjectives occurs with every noun can be used as a measure of meaning difference. For it is not merely that different members of the one class have different selections of members of the other class with which they are are actually found. More than that: if we consider words or morphemes A and B to be more different than A and C, then we will often find that the distributions of A and B are more different than the distributions of A and C. In other words, difference in meaning correlates with difference in distribution.*

A great illustration of capturing the meaning of the word from linguistic context is given by Stefan Evert[1]:

Example 3.2.1 (*Bardiwac*)

- ◇ He handed her a glass of *bardiwac.*
- ◇ Beef dishes are made to complement the *bardiwacs.*
- ◇ Nigel staggered to his feet, face flushed from too much *bardiwac.*
- ◇ Malbec, one of the lesser-known *bardiwac* grapes, responds well to Australia's sunshine.
- ◇ I dined off bread and cheese and this excellent *bardiwac.*
- ◇ The drinks were delicious: blood-red *bardiwac* as well as light, sweet Rhenish.

Even without any previous intuition about the meaning of a fictional word 'bardiwac' we can easily capture the meaning of it from the context. After reading the sentences from the Example 3.2.1 we can easily understand that 'bardiwac' is a heavy red alcoholic beverage made from grapes.

Distributional semantic model can be defined as following:

[1]http://www.stefan-evert.de

Definition 3.2.1

A distributional semantic model (DSM) is a co-occurrence matrix **M***, such that each row* **x** *represents the distribution of a target term across contexts where the target term appears.*

Table 3.1 presents a toy example of a co-occurrence matrix, where rows represent target terms and columns represent contexts. Here, contexts directly characterised by word counts, for example, in each sentence that contains target terms. A feature vector of a word 'boat' would be $x_{\vec{boat}} = (42, 30, 0)$.

	run	*bark*	*cuddle*
dog	45	35	65
boat	42	30	0
cat	36	1	85

Table 3.1 Toy co-occurence matrix

Semantic similarity between words can be predicted from the DSM as proximity between the corresponding feature vectors. One widely used measure is the cosine of the angle θ between the two vectors $\vec{a}$ and $\vec{b}$:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum\limits_{i=1}^{n} a_i b_i}{\sqrt{\sum\limits_{i=1}^{n} a_i^2} \sqrt{\sum\limits_{i=1}^{n} b_i^2}}, \tag{3.1}$$

where a_i and b_i are components (contexts) of vectors $\vec{a}$ and $\vec{b}$ respectively. The value 0 indicates absence of semantic similarity and 1 indicates full semantic similarity.

According to our toy example cosine similarity between the words 'dog', 'boat' and 'cat' can be computed:

	dog	*boat*	*cat*
dog	1	0.66	0.90
boat	0.66	1	0.32
cat	0.90	0.32	1

Table 3.2 Cosine similarity for toy DSM

According to toy example, the cosine similarity between ‘dog’ and ‘boat’ is 0.66 and the cosine similarity between ‘dog’ and ‘cat’ is 0.90, which means that ‘dog’ is more similar to ‘cat’ than to ‘boat’.

It is important to mention that real examples of DSMs with word counts might be very sparse, producing matrix of hundreds of thousands cells with zero value. This would lead to manageability and high computational complexity of operations with such matrix. In practice, various dimensionality reduction techniques are used, such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD), etc.

3.2.3 Psychological and Linguistic Viability of DSMs

DSMs show good results in modelling psychological and linguistic phenomena. In this section we are going to present state-of-the-art results showing how DSMs approximate human performance in various tasks.

TOEFL synonym identification task The TOEFL is an obligatory test for foreign students who would like to study at a university in an English speaking country. One of the task contains 80 items with candidate synonyms, e.g. presenting a choice of four candidates for the word ‘urgently’: ‘typically’, ‘conceivably’, ‘tentatively’ or ‘desperately’. On average, for this task the performance of the native speakers was 97.75%, whereas the performance of the non-native speakers was 86.75% (Rapp, 2004). Models based on distributional hypothesis are able to reach 100% accuracy on this test set, namely Principal Component vectors with Caron P (PCCP) model that was introduced by Bullinaria and Levy (2012).

RG-65 semantic similarity judgment task Rubenstein and Goodenough (1965) (RG-65) data set contains 65 pairs of nouns that were rated by 51 subjects that gave a score from 0 to 4 to each pair (see Example 3.2.2). With DSMs, for each pair of words, feature vectors are taken and then cosine similarity can be measured. Resulting similarities can be checked for correlations with RG-65 data set using Pearson’s r correlation coefficient, with a value between $+1$ and -1 inclusive, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. Padó and Lapata (2007) use dependency-based DSM, that takes semantic relations into account. Their model shows strong correlations with human judgements ($r = 0.8$).

Example 3.2.2 (*Judgments from RG-65 task*)

noon	string	0.04
bird	cock	2.63
gem	jewel	3.94

Semantic priming There is evidence from a range of experimental paradigms that the effort that is required for processing a word is influenced by the context where that word occurs. This phenomena is called *contextual facilitation* and is caused by various forms of *semantic priming.* A task can include taking a decision whether a target word is a word or not, primed with stimulus word that is in certain semantic relation with the target word. For example, subjects can perform better with reading or recognizing the word 'avocado' after they encountered the word 'apple'. DSM-based experiments are using cosine similarity to distinguish related and unrelated primes for target word. McDonald and Brew (2004); Padó and Lapata (2007) show significant effects for various semantic relations between a prime and target word.

Language learning and acquisition DSMs achieve good performance using only large amount of linguistic data as learning input that is similar to the input received by human learners. Though, language acquisition theorists as Bloom (2001) emphasize the importance of interaction and attention in the language acquisition, they recognize that the vocabulary size that teenagers command by end of high-school (in the order of tens of thousands of words) can only be acquired by bootstrapping from linguistic data. This bootstrapping is similar to distributional models, as we learned the meaning of the word 'bardiwac' from sentences presented in Example 3.2.1. However, humans are often offered a single exposure to a word in context to acquire its meaning, and there are no studies that systematically evaluate the quality of DSMs to model single occurrences of the word (Baroni et al., 2014).

Neuroscience There is support from neuroscience for the view that concepts can be represented as neural activation patterns over broad areas (Haxby et al., 2001), and can be naturally encoded as vectors. Murphy et al. (2012) shows how corpus-based methods can predict brain activation pattern while subject is thinking of a concept, which can lead to quite speculative hypothesis of connection between DSM and structure of concept encoding in human brain.

Grounding counterargument There is an important argument against DSM, saying that representing a meaning of the word just in terms of other symbols lacks *grounding* in sensory-motor system and thus lacks connection with external word. Chinese Room thought experiment (Searle, 1988) is a traditional argument that is given against symbolic models. Also, there is a theory of *embodied cognition* that proposes that concepts are organized in the brain, based on their sensory and motor properties, and that simulation of actions and perceptions in the brain plays a central role in cognition (Barsalou, 2010). However, it seems that not all the concepts are obviously presented in external world, such as 'truth' or 'politics', that suggests that both embodied cognition and relation between symbols can play an important role.

3.2.4 Neural Word Embeddings

Notorious and widely applied models that learn vector representations by using recurrent neural network (RNN) were proposed by Mikolov et al. (2013). In Section 3.3.5 I will briefly introduce the notion of recurrent neural network.

RNN is trained with back-propagation that adjusts the word vectors by walking through huge corpus of texts. They are often referred as *word2vec* (by the name of a tool, provided by the authors) or *word embeddings*; and, use one of two model architectures to produce a distributed representation of words: continuous bag-of-words or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words.

Thought the model itself doesn't have any knowledge of morphology or semantics, it provides interesting results on both semantic and syntactic tests. Test sets were composed of analogy questions of the form "*a* is to *b* as *c* is to _?" testing different syntactic and semantic relations.

Syntactic tests contain examples like "*see* is to *saw* as *return* is to _" (the system should reply with the word *returned*). On the other hand semantic test contain also analogy questions like "*clothing* is to *shirt* as *dish* is to _" (the system should reply with the word *bowl*).

Mikolov et al. (2013) showed that these relations can be obtained using a vector offset model denoting which pairs of words share a particular relation according to the offset between their vector representations. Figure 3.1 depicts how vector offset can be represented in 2-dimensional space.

Mikolov et al. (2013) claim that RNN language model and vector offset model together provide good results on capturing semantic and syntactic regularities. Namely they showed almost 40% correct matches on syntactic dataset and outperformed previous state-of-the-art on the semantic similarity task of SemEval-2012.

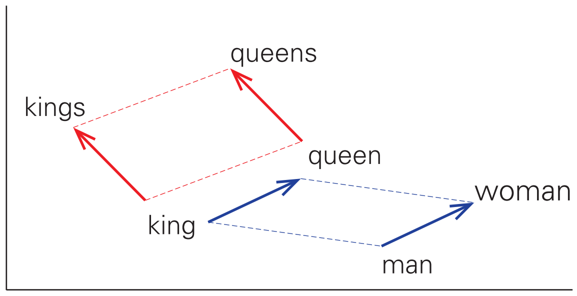

Fig. 3.1 Syntactic (shown in red) and semantic (shown in blue) regularities in vector space. Adapted from Mikolov et al. (2013).

3.2.5 Compositionality of DSM

As we have distributional semantic concepts that are showing great results in different tasks, it is important to point the ambitious aim of obtaining a general-propose semantics for natural language. Current approaches use logic to represent sentence semantics, which has little traction in capturing word similarity that can be obtained using distributional semantics.

Approaches on composition of sentence meaning propose different vector composition techniques. For example, Grefenstette and Sadrzadeh (2011) represent subject and object verbs as vectors, combining them using tensor product and use component-wise multiplication to matrix that represents transitive verbs. The meaning of a phrase "subj verb obj" is computed as for example $(\vec{john} \otimes \vec{duck}) \odot \vec{see}$ for the sentence "John sees ducks".

There are many open problems for sentence semantics. One of these problems is the scaling problem that raise a following question: how fixed-length vectors can provide a representation for sentence of arbitrary length and structure in order to obtain fine-grained sentence similarity. Another problem is about representing function words, such as "not": how should they be encoded in vector space to adjust the meaning of a constituent (Erk, 2012).

3.3 Deep Learning

In recent years deep learning computational techniques dramatically improved state-of-the-art in such areas as computer vision, speech recognition, genomics, etc. Deep learning models include multiple processing layers that support different abstractions of the data.

The main difference between deep learning techniques and conventional machine learning is that the latter requires handcrafted feature extraction from the raw data. Typically, input is transformed into certain representation (i.e. feature vectors). For example, computer vision system will receive input image as an array of pixel colour values and extract other additional handcrafted features. These features will be passed to a classifier, that will learn important features with respect to the task (e.g. distinguishing cars from houses) and classify patterns in the input.

Deep learning methods automatically learn multiple levels of representation, each one is slightly more abstract than the previous one. For example, for computer vision deep learning system will receive as input a raw image given as a matrix with pixel colour values. The first layer will detect which edges are present in a picture and how are they inclined. The next layer will detect how these edges are connected together. The next layer will represent a more abstract features and so on. Important feature of deep learning systems is that these representational layers are not created by humans, but they are learned from data.

The main ideas behind deep learning came from the 1970s and earlier but it took time to spread. Key drivers that led to deep learning emergence are: (1) consistent growth of computational power including GPUs (graphical processing units) and (2) availability and ubiquity of huge amounts digital data (i.e. texts and images) in digital form that emerged with the tremendous growth of the Internet.

In this section I will describe key ideas behind deep learning and briefly review main architectures that are most important for solving natural processing tasks.

3.3.1 Machine learning

Learning from data performed by computer was defined by Mitchell (1997) in the following way:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Machine learning tasks T typically describe, how machine should process a given example, which is usually represented as a collection of features. There is a wide variety of tasks that could be solved using machine learning approaches: classification, numerical value prediction, transcription (i.e. speech-to-text), machine translation, anomaly detection, synthesis, etc. In the present work, the machine learning task can be defined as the following: "Given two questions as examples, discover if they are semantically equivalent".

Performance measure P is usually specific for the task that is conducted by the system. The typical measures are accuracy and error rate, which are proportions of correct or incorrect outputs respectively. Error rate, or loss can be measured as a binary or a continuous-valued score. For some tasks, such as machine translation, measures should be more fine-grained and the selection of such measures is an important research issue.

Important condition for measuring the performance of a model is its evaluation on unseen data. This would show how well the model will perform when deployed in the real world. For this reason performance is measured on a test set that is separated from the one used for training the system.

In terms of the *experience* that the system is allowed to get during training, machine learning techniques can be split into two broad categories: *supervised* and *unsupervised.*

A *supervised learning* algorithm operates with examples which are associated with a certain label. For example, training dataset of questions is tagged with labels that determine if the questions in the pair are semantically equivalent.

In contrast, *unsupervised learning* systems handle examples that are not labelled and are learning to extract some sense without any guidance. For example, unsupervised systems of network security domain can be used to detect patterns and anomalies in network traffic.

3.3.2 Inspirations from neuroscience

One of historical names for deep learning is *artificial neural networks* (ANNs). This name came from the loose inspiration that computer scientists and engineers received from the biological brain structure. Neural networks were serving as models that were used to understand how human or animal brain functions.

Another inspiration from neuroscience is a reason to hope that lots of different tasks can be solved by a single algorithm that will operate through deep neuron structure. The studies on ferret's brain shown that if certain areas of their brain would be rewired to receive visual signals, they will learn to "see" with a brain area that is different from the visual cortex (Von Melchner et al., 2000) .

Despite of these inspirations, we should take into account that neurons compute functions that are different from those functions computed by units used in artificial neural networks. Making this functions closer to biological reality does not lead to any performance improvements of deep learning algorithms (Goodfellow et al., 2016). Also, while several neural network architectures were inspired by neuroscience, we do not have yet enough evidence about biological learning. So it is not yet practical to use neuroscience to provide guidance for the learning algorithms that are used to train deep learning architectures.

The main difference between deep learning algorithms research and computational neuroscience is that the latter is aimed to build accurate models of how the brain works, while deep learning research deals with building computer systems that would solve tasks that require intelligence with the most favourable result.

3.3.3 Deep feedforward neural networks

Feedforward neural networks are the most basic and expressive deep learning models. Feedforward networks that are used for classification purposes maps an input $\mathbf{x}$ to a category y approximating some function f. The resulting best function approximation can be defined as mapping $y = f(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of parameters which provide this approximation.

This kind of model is a network because it is constructed by chain composition of layer of functions, e.g. $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ that are used to approximate function $f(x) = f^{(3)}(f^{(2)}(f^{(1)}))$. Each of the functions represents a *layer* of a network. The last layer is called *output layer* and the others are called *hidden layers* because training data does not show the desired output for each of them.

The term *feedforward* means that there are no feedback connections in which outputs of layers are connected back into the same or bottom layer.

For computing the values for each of the layers various activation functions (typically non-linear) are used. One of the most important functions for computing hidden layer values is *linear rectifier function* $g(z) = max\{0, z\}$

The values of hidden layer $\mathbf{h}$ are computed using the following transformation:

$$\mathbf{h} = g(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{3.2}$$

where W provides the weights of a transformation of input $\mathbf{x}$ into $\mathbf{h}$ and $\mathbf{c}$ provides the biases. Another popular activation functions are sigmoid $\sigma(z)$ and hyperbolic tangent $tanh(z)$. Output layer values, denoted as $\hat{\mathbf{y}}$, are computed in a similar fashion.

The forward flow of information from input $\mathbf{x}$ to the series of hidden units up to the final value $\hat{\mathbf{y}}$ is called *forward propagation*. The scalar cost $J(\boldsymbol{\theta})$ produced by the network represents its current performance measure. The *backpropagation* (Rumelhart et al., 1988), an abbreviation for *backward propagation of errors* calculates the gradient of a cost function with respect to all the weights in the network. Gradient is used to update the weights and minimize the cost improving the performance of the network for the next training examples.

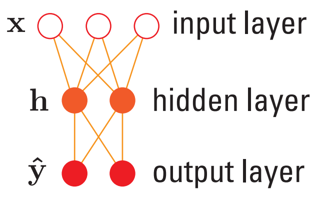

Fig. 3.2 Feedforward fully-connected neural network.

Figure 3.2 shows a simple example of a *feedforward fully-connected neural network*. It contrasts with recurrent neural network because it does not have any feedback connections. The feature of a network being *fully-connected* meaning that each of the layer units is connected to each of the units of the adjacent layer(s) contrast with convolutional neural network. Convolutional neural networks and recurrent neural networks will be briefly explained in the next sections.

3.3.4 Convolutional neural networks

Convolutional neural networks (CNNs) (LeCun et al., 1989) are a subset of feed-forward artificial neural network in which the neurons connectivity is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the receptive field. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be mathematically approximated by a convolution operation. Key ideas behind CNNs are: sparse connectivity, shared weights, pooling and the use of many layers.

CNNs have *sparse connectivity* between layers, for example the input image might have thousands of pixels and meaningful features are extracted not from individual pixels but from the regions that occupy certain patch of an image (e.g. 10×10 pixel

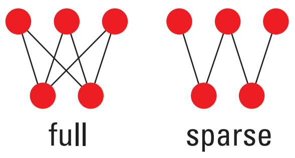

Fig. 3.3 Fully-connected layers vs sparsely connected layers.

area). Patches of an image are connected to the set of network units called *feature maps*. Convolutional layers stack together such that each convolutional layer apply certain set of feature maps. All units in a feature map share the same set of weights called *filter bank*. For one layer different feature maps with correspondent filter bank can exist.

A pooling layer is used to merge similar features into one and typically it computes the maximum of a local patch into one or several feature maps (however, in the work by Bogdanova et al. (2015), they sum up the local patch, see Section 4.1.3).

Convolution neural networks are deep: two or three stages of convolution and pooling are stacked, followed by more convolutional and fully-connected layers.

CNNs brought huge revolutionary success in computer vision and are important for various recognition and pattern detection tasks, such as traffic sign recognition, the segmentation of biological images for the detection of faces, text, pedestrians and human bodies in natural image (see LeCun et al. (2015) for the collection of papers that reach or overcome the current state-of-the-art in various tasks).

Recent CNN architectures have 10 to 20 layers of non-linear rectifier units, hundreds of millions of weights, and billions of connections between units. Training such large networks could have taken weeks in 2013, but progress in hardware, software and algorithm parallelisation have reduced training times to a few hours in 2015 (LeCun et al., 2015).

Section 4.1 of the next chapter will describe a particular architecture of CNN that was used in the present work for the task of detecting semantically equivalent questions.

3.3.5 Recurrent neural networks

Recurrent neural networks (RNNs) are notorious for their utilisation in tasks that involve sequential input such as speech and language.

Hidden layers of RNNs represent discrete time steps (states) of the network where each subsequent state receives an input from the preceding state. This can help to predict the next word given previous words, for example, in statistical language modelling. Figure 3.4 shows two ways to represent RNN: folded (left) and unfolded (right) computational graph. Hidden state s_t keeps a summary of aspects of past sequence that are relevant to the task.

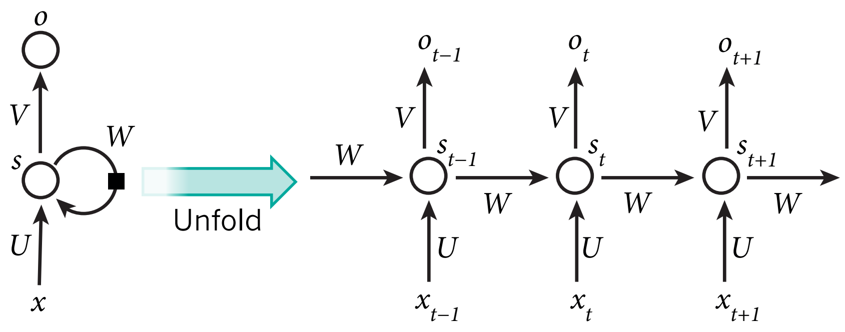

Fig. 3.4 Folded and unfolded recurrent neural network representation (from LeCun et al. (2015). Input is denoted as x, output—as o, weight matrices U, V, W are shared between time steps.

In more sophisticated architectures, such as LSTM (long short-term memory) (Hochreiter and Schmidhuber, 1997), this summary of aspects can selectively keep only the most relevant aspects from previous time steps. This kind of networks have a memory cell, a special unit that acts like an accumulator: it has a self-loop that can produce path where the gradient can flow for long duration.

The LSTM got a remarkable success in many applications that involve sequence modelling: handwritten recognition and generation, speech recognition, machine translation, image captioning, etc (see LeCun et al. (2015) for the collection of papers that reach or overcome the current state-of-the-art).

3.4 Summary

This chapter has introduced the main background notions that will be used to tackle the task that was defined in Chapter 2. Notions that has been considered include distributional semantics and deep learning along with the presentation of the state-of-the-art approaches to the task of question answering. Next chapter will continue with the explanation how the system was implemented and what result has been obtained.

Chapter 4

Implementation and evaluation

In this chapter, I describe how the experiment aimed to replicate the work by Bogdanova et al. (2015) was carried out.

In Section 4.1, I will start with describing the architecture of convolution neural network (CNN) that was used for the experiment. Section 4.2 will describe how the CNN was implemented. Section 4.3 will show the result of evaluating the implemented model together with the results obtained by Bogdanova et al. (2015) which will conclude the part of dissertation that is related to the replication exercise.

Sections 4.4 and 4.5 cover two contributions that were added to the dissertation, by changing the way the text was preprocessed and by changing the values of initial word embeddings, respectively.

4.1 Convolutional neural network for the detection of semantically equivalent questions

In this section, I describe the architecture of a convolutional neural network (CNN) to be replicated. The architecture was introduced by Bogdanova et al. (2015) and it is based on the previous work by dos Santos and Gatti (2014) for representing text segments and by dos Santos and Zadrozny (2014) for representing words on a character-level. The CNN takes two questions as an input, forms representations of each of them, and compares the representations using cosine similarity measure.

Figure 4.1 shows layers of the CNN: word representation layer (WR), convolution layer (CONV), pooling layer (POOL) and cosine similarity measurement layer.

Forming of question representation falls into 3 main steps:

1. forming word representations

2. applying convolutional filter
3. pooling the product of convolution filters

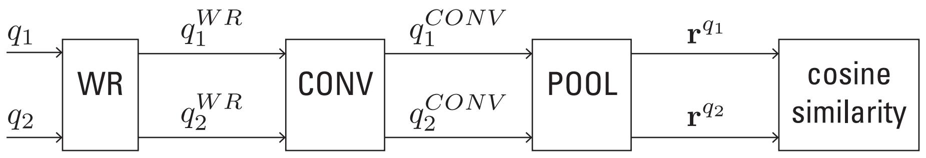

Fig. 4.1 CNN architecture.

4.1.1 Word representation layer

Word representations are real-valued vectors of size d denoted as $\mathbf{r}_{w_i} \in \mathbb{R}^d$. Initial values for word representations are obtained using *word2vec* tool (Mikolov et al., 2013); reasons for building word representations with the tool were described in Section 3.2.4, and in Section 4.2.3 I will describe how the word representations were created.

Word representations are loaded into an *embedding matrix* $\mathbf{W}^{WR} \in \mathbb{R}^{d \times |V|}$, where V is a fixed-sized vocabulary and each column i of the embedding matrix is a vector $\mathbf{r}_{w_i}$ that represents the i-th word in the vocabulary.

In order to get a word representation a matrix-vector product is used:

$$\mathbf{r}^{w_i} = \mathbf{W}^{WR}\mathbf{v}^{w_i} \tag{4.1}$$

where $\mathbf{v}^{w_i}$ is a *one-hot encoded* word, that is a vector that has value 1 at position i and zeros at all other positions. An input to CNN for each question is a sequence of N one-hot encoded words:

$$q = \{\mathbf{v}^{w_1}, \mathbf{v}^{w_2}, \ldots, \mathbf{v}^{w_N}\} \tag{4.2}$$

And the word representation layer of a network produces a set of word representations:

$$q^{WR} = \{\mathbf{r}^{w_1}, \mathbf{r}^{w_2}, \ldots, \mathbf{r}^{w_N}\} \tag{4.3}$$

where $\mathbf{r}_{w_i} \in \mathbb{R}^d$ and N is the number of words in a question.

The size of word embedding d is a hyper-parameter to be set while training the initial values for the word representation layer. The values of the embedding matrix $\mathbf{W}^{WR}$ are updated using the backpropagation algorithm while new pairs of questions are presented to the network during the training procedure.

4.1.2 Convolution layer

The convolutional layer receives a set of vectors q^{WR} as an input and applies convolutional filter to each k-gram of the input tokens. Vectors in each k-gram are concatenated in order to produce k-gram vector $\mathbf{z}_n \in \mathbb{R}^{dk}$ that is centralized in n-th word of input sequence. Out-of-range input values r^{w_i} where $i < 1$ or $i > N$ are taken to be zero.

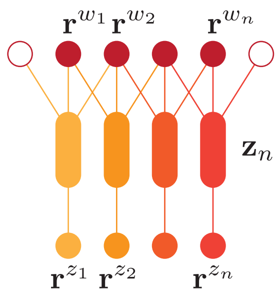

Fig. 4.2 Convolutional layer (for $k = 3$). Out-of-range tokens are shown as empty circles.

Figure 4.2 represents how convolutional layer manipulates the input. Convolutional layer is applied in the following steps: (i) Taking a dot product of a weight matrix $\mathbf{W}^{CONV}$ with each concatenated k-gram; (ii) Adding bias vector $\mathbf{b}^{CONV}$; (iii) Applying hyperbolic tangent function.

Resulting convolutional layer output is a set of vectors q^{CONV}:

$$q^{CONV} = \{\mathbf{r}^{z_1}, \mathbf{r}^{z_2}, \dots, \mathbf{r}^{z_N}\} \tag{4.4}$$

where each $\mathbf{r}^{z_n} \in \mathbb{R}^{cl_u}$ is computed using the following equation:

$$\mathbf{r}^{z_n} = f(\mathbf{W}^{CONV}\mathbf{z}_j + \mathbf{b}^{CONV}) \tag{4.5}$$

where $\mathbf{W}^{CONV} \in \mathbb{R}^{cl_u \times dk}$ is a weight matrix, $\mathbf{b}^{CONV} \in \mathbb{R}^{cl_u}$ is a bias vector, and f is a hyperbolic tangent function. Values of the weight matrix $\mathbf{W}^{CONV}$ and bias vector $\mathbf{b}^{CONV}$ are updated using backpropagation algorithm while new pairs of questions are presented to the network during the training procedure (detailed in Section 4.1.5 below). The length of k-gram and the size of convolutional filter cl_u are hyper-parameters to be set prior to training the CNN.

4.1.3 Pooling layer

Pooling layer combines representations of each k-gram into representation of a whole question $\mathbf{r}^q$. Pooling layer operates by summing up the k-gram representations and by applying hyperbolic tangent function to the sum:

$$\mathbf{r}^q = f\Big(\sum_n \mathbf{r}^{z_n}\Big) \tag{4.6}$$

where $\mathbf{r}^q \in \mathbb{R}^{cl_u}$ is a representation of a question, $\mathbf{r}^{z_n} \in \mathbb{R}^{cl_u}$ is a representation of each k-gram, and f is a hyperbolic tangent function.

Note that the dimensionality of a question representation cl_u is a hyper-parameter that is independent of the size of a question. Also, it is important to mention that pooling layer can use other functions than element-wise sum, i.e. element-wise maximum.

4.1.4 Measuring the question similarity

Preceding layers of CNN transform two questions q_1 and q_2 into representations $\mathbf{r}^{q_1}$ and $\mathbf{r}^{q_2}$. For any two questions, CNN uses the same hyper-parameters (k and cl_u) and the same weight and bias tensors ($\mathbf{W}^{WR}$, $\mathbf{W}^{CONV}$ and $\mathbf{b}^{CONV}$).

Two questions are compared using cosine similarity measure between respective question representations:

$$s(q_1, q_2) = \frac{\mathbf{r}^{q_1} \cdot \mathbf{r}^{q_2}}{\|\mathbf{r}^{q_1}\|\|\mathbf{r}^{q_2}\|} \tag{4.7}$$

The resulting similarity should be from 1 for equivalent questions and 0 for non-equivalent questions.

4.1.5 Training procedure

The CNN parameters $\mathbf{W}^{WR}$, $\mathbf{W}^{CONV}$ and $\mathbf{b}^{CONV}$ are denoted as $\boldsymbol{\theta}$ for current training step. The CNN is trained in order to minimize the mean-squared error with respect to training set D:

$$\sum_{(q_1,q_2,y)\in D} \frac{1}{2}(y - s_{\boldsymbol{\theta}}(q_1, q_2)) \tag{4.8}$$

where $s_{\boldsymbol{\theta}}(q_1, q_2)$ is a cosine similarity between questions q_1 and q_2 for current CNN parameters $\boldsymbol{\theta}$ and y is a correct label for the question pair: 1 for semantically equivalent questions and 0 for non-equivalent questions.

Mean-squared error is minimized using stochastic gradient descent (SGD). Gradients of θ are computed using backpropagation algorithm.

4.2 Implementing the CNN

4.2.1 Data preprocessing

In order to prepare all text data to be used as an input for training the CNN and the initial word representations, the following text preparation procedures took place:

1. Image removal
2. URL removal
3. `<code>` block removal
4. Text tokenisation
5. Lowercasing all the tokens

The `<code>` tag a in question text is usually used to insert snippets of the code in some programming language that could contain words with different meanings, e.g. word *for* in a question text might denote a preposition that contrasts with its very specific meaning inside a `<code>` block, which supposedly denotes for-loop. Bogdanova et al. (2015) shows better results with `<code>` block removed. Accordingly, `<code>` block was removed for all the experiments here.

For English text tokenisation, I used Stanford Tokenizer (Manning et al., 2014) instead of Natural Language Toolkit (NLTK) used by Bogdanova et al. (2015) because my pipeline for creating the initial word representations was Java-based, and NLTK was only available for Python.

Other procedures replicate the ones used by Bogdanova et al. (2015).

4.2.2 Corpora parameters

For the experiments reported here, I took slightly different data sets given that the data sets used in (Bogdanova et al., 2015) were not available at this time. I used AskUbuntu and Meta StackExchange dumps from September 2014 instead of the May 2014 version used by Bogdanova et al. (2015). Table 4.1 shows main differences between the datasets I used and the datasets used in the reference work to be replicated.

Community	*Dump date*		*Millions of tokens (for DSM)*	
	This work	*Reference*	*This work*	*Reference*
AskUbuntu	Sept 2014	May 2014	38	121
META	Sept 2014	May/Sept 2014[a]	19	19

Table 4.1 Corpora characteristics, where reference is the dataset used in Bogdanova et al. (2015).

[a]For training the CNN Bogdanova et al. (2015) used September 2014 data dump.

4.2.3 Building and training the DSM

The corpora that I used for creating the initial word representation contains 38 millions tokens from the AskUbuntu dump and 19 million of tokens from the META StackExchange dump. Table 4.2 shows characteristics of the initial word representations and corpora that were used for training them.

Dump	*Vector size (d)*	*Total types*	*Total tokens*
AskUbuntu	200	68 K	38 M
META	200	30 K	19 M

Table 4.2 DSM parameters for each corpus.

I used DeepLearning4j[1] toolkit for creating the initial word representations. Bogdanova et al. (2015) specify only the skip-gram neural network architecture and the embeddings dimensionality of 200 as training parameters for their best run. Hence, in my work, besides these parameters, all the other hyper-parameters were taken from vanilla version of *word2vec* algorithm implemented in DeepLearning4j toolkit[2].

I have done basic tests in order to check the adequacy of the word embeddings that were created. Given that the datasets include domain-specific questions, thus instead of a general test (Mikolov et al., 2013) like $\mathbf{x}_{king} - \mathbf{x}_{man} + \mathbf{x}_{woman} \approx \mathbf{x}_{queen}$

[1]http://deeplearning4j.org

[2]http://deeplearning4j.org/word2vec

another test was picked. This test requires the system to rate all the representations of the words in the vocabulary by their respective proximity (designated by cosine distance) to this vector $\mathbf{x}$:

$$\mathbf{x} = \mathbf{x}_{likes} - \mathbf{x}_{like} + \mathbf{x}_{contain} \tag{4.9}$$

Successful pass of this test assumed that the representation of the word 'contains' should be rated as the most proximate representation to $\mathbf{x}$.

This test was passed for the word embeddings that I generated for both domains: AskUbuntu and META StackExchange.

4.2.4 Building and training the CNN

For building the CNN, I extracted from the corpora pairs of questions tagged with duplicate and with non-duplicate label. Table 4.3 shows sizes of training, validation and test sets that were used for each corpus. These sizes replicate the ones used by Bogdanova et al. (2015).

In each set, the pairs of questions were split in subsets of duplicates and non-duplicates in a balanced way (50% duplicates and 50% non-duplicates).

Community	*Total pairs*	*Duplicates*	*Training*	*Validation*	*Test*
AskUbuntu	167765	17115	24000	1000	6000
META	67746	19456	20000	1000	4000

Table 4.3 Sizes of training, validation and test sets for each corpus.

I used Keras (Chollet, 2015) Python library with Theano (Al-Rfou et al., 2016) back-end for building and training the CNN. Table 4.4 shows hyper-parameters that were used. The last two parameters were not explicitly mentioned in the reference paper. They were set to the values shown in the table. Other parameters are taken from vanilla CNN architecture as it was implemented in Keras[3] and Theano[4] libraries. Mean-squared error computation and backpropagation algorithm are also implemented by those libraries.

[3]https://github.com/fchollet/keras/releases/tag/1.1.0

[4]https://github.com/Theano/Theano/releases/tag/rel-0.8.2

Parameter	*Value*	*Description*
d	200	Size of word representation
k	3	Size of k-gram
cl_u	300	Size of convolutional filter
λ	0.005	Learning rate
batch size	1	Number of examples per gradient update
epochs	20	Number of training epochs

Table 4.4 CNN hyper-parameters. Upper part of the table shows parameters explicitly mentioned by Bogdanova et al. (2015).

4.3 Replication

For the validation and testing, I used the CNN that was trained on the full training set (24K for AskUbuntu and 20K for META StackExchange). Table 4.5 shows evaluation results: validation accuracy (for the best epoch) and testing accuracy. Results that are obtained by the reference paper (Bogdanova et al., 2015) are shown in the first two lines.

	Community	*4K val.*	*Full val.*	*Full test*
Reference work	AskUbuntu	92.4	93.4	92.9
	META	–	92.8	92.7
Present work	AskUbuntu	91.8	92.3	94.1
	META	–	96.1	94.2

Table 4.5 CNN accuracy on the validation and test sets.

4.4 Impact of text preprocessing

From the work of Bogdanova et al. (2015) it is unclear how do the authors treat mentions to possible duplicate questions. These mentions occur in the body of a question which is considered a possible duplicate of another question. Below there is an example of the HTML structure that is present in the body of such question.

```
<blockquote><p>
```

```
  <strong>Possible Duplicate:</strong><br>
  <a href="http://askubuntu.com/questions
           /86164/how-do-i-fix-flash-issues">
    How do I fix Flash issues?
  </a>
</p></blockquote>
```

As can be seen from the example above, there is a remark ‘Possible duplicate’ that this question is a possible duplicate of some other question. Also, there is a link whose text exactly reproduces the title of a question, which corresponds to the possible duplicate one. A remark and a link like these are present in the majority of question pairs that are marked as duplicates cross-referencing the element in a given pair.

I hypothesised that this could had a significant impact on the results obtained by Bogdanova et al. (2015). In order to validate this hypothesis, I made two runs with different text preprocessing on AskUbuntu corpus: in the first one the duplicate quotes were kept; in another one, they were removed. The CNN was trained using 4K balanced training set and was validated over 1K balanced set.

Table 4.6 presents the difference between the results obtained with respect to the validation accuracy reported by Bogdanova et al. (2015). Scores for validation and testing accuracies obtained by keeping duplicate mentions are very close to the reference ones. This may indicate that Bogdanova et al. (2015) did not remove mentions of duplicate questions. The run with removed duplicate quotes shows the much lower testing accuracy, e.g. 73.3% (against 94.1%) for AskUbuntu.

These scores are more in line with the state of the art. For example, the best systems participated in SemEval 2016 community question answering shared task (Nakov et al., 2016) reached the highest accuracy of 79.43% for the Subtask B that was intended to evaluate how good do the models perform in measuring the similarity between two questions.

Duplicate quotes	*4K AU val.*	*AU full val.*	*AU test*	*META val.*	*META test*
kept	91.8	92.3	94.1	96.1	94.2
removed	71.8	73.8	73.3	57.3	55.7
Bogdanova et al. (2015)	92.4	93.4	92.9	92.8	92.7

Table 4.6 Comparison of runs over the validation and test sets. AU—AskUbuntu. The best validation scores are shown.

4.5 Impact of word embeddings

Bogdanova et al. (2015) show that in-domain word embeddings, i.e. trained on AskUbuntu community data are more beneficial than word embeddings with larger volume of out-of-domain training data, i.e. trained on English Wikipedia.

Additionally, I wanted to study how the model would perform without a pre-training of word embeddings, in particular with just random initialisation of them using uniform distribution.

Table 4.7 shows that the evaluation results for randomly initialised word embeddings outperform the ones with pre-trained in-domain word embeddings.

Word embeddings	*4K AskUbuntu val.*	*AskUbuntu full val.*	*AskUbuntu test*
pre-trained	71.8	73.8	73.3
randomly initialised	73.3	76.9	74.5

Table 4.7 Comparison of runs over the validation and test sets. The best validation scores are shown.

4.6 Summary

This chapter has presented the part of work related to the task of question answering and to the replication of the results that were obtained by Bogdanova et al. (2015). I have started with a detailed description of the convolutional neural network (CNN) used in the experiment. Then, I have described how the CNN was implemented, including how the corpora were prepared, and how the distributional semantic model and the CNN were trained. Results that have been presented are comparable or better than the ones obtained by Bogdanova et al. (2015).

Another part of this chapter has been related to two key contributions of the dissertation. The first contribution consisted in tackling a possible drawback in the work done by Bogdanova et al. (2015) regarding the preparation of the text, which may have left untouched certain clues that could guide the system in detecting the duplicate question. Taking away these clues in our experiments has lead to a drop in performance of the system.

The second contribution was related to the evaluation of the impact of pre-trained word embeddings on system performance. The pre-trained word embeddings has been replaced by randomly initialised ones, and as a result performance of the system has improved.

This chapter closes the part of the work related the replicability of the results of the work done by Bogdanova et al. (2015) and also concludes the work related to the task of question answering.

In the next chapters I will discuss the impact that can be brought by convolutional neural network models into other tasks and languages other than English, namely entailment recognition for Portuguese and paraphrase detection for Russian.

Chapter 5

Portuguese Entailment Recognition Task

In this chapter, I show how the convolutional neural network model that has been described in the previous chapter can be evaluated on a task from another domain, namely entailment recognition for Portuguese.

The task itself will be defined in Section 5.1, Section 5.2 will discuss the results of evaluation of the system and in Section 5.3, I will describe the directions of further improvements that might be taken into account.

5.1 Task definition

The Evaluation of Semantic Similarity and Textual Inference for Portuguese shared task (ASSIN, "Avaliação de Similaridade Semântica e Inferência Textual") (Fonseca et al., 2016) was co-located with the PROPOR-2016 conference and consisted of two tasks: semantic similarity detection and entailment recognition task. In the present chapter, I will describe the experiment of applying the model that was described in Chapter 4 to the task of entailment recognition of ASSIN.

In pragmatics (subfield of linguistics), *entailment* is a relationship between two sentences (A and B), where the truth of one (A) requires the truth of the other (B). Here is an example of entailment from the ASSIN corpus:

(1) A. Como não houve acordo, a reunião será retomada nesta terça, a partir das 10h.

As there was no agreement, the meeting will resume this Tuesday, starting at 10h.

B. As partes voltam a se reunir nesta terça, às 10h.

The parties will meet again this Tuesday at 10h.

When entailment is mutual, the relation between sentences A and B is called *paraphrase*. In this case, two sentences are semantically equivalent. An example of paraphrases from the ASSIN corpus is provided below.

(2) A. Vou convocar um congresso extraordinário para me substituir enquanto presidente.

I am going to call for a special congress to replace me as a chairman.

B. Vou organizar um congresso extraordinário para se realizar a minha substituição como presidente.

I am going to organize a special congress to accomplish my replacement as a chairman.

The ASSIN corpus also includes pairs of sentences not related by entailment or paraphrase such as:

(3) A. As apostas podem ser feitas até as 19h (de Brasília).

The bets can be made up to 19h (BRT).

B. As apostas podem ser feitas em qualquer lotérica do país.

The bets can be made in any lottery of the country.

The ASSIN task requires that input pairs of sentences should be classified into the aforementioned three classes (viz. entailment, paraphrase, unrelated). The model can only provide binary classification, so I will handle only the task of entailment recognition without further recognition of a paraphrase. This means that pairs of sentences like (1) and (2) are treated as one class, and sentences like (3) — as another class.

In this case, a cosine similarity score of 1 between two sentences means that these sentences are under entailment or paraphrase relation and a score of 0 means that these two sentences are unrelated.

5.2 Evaluation and results

The ASSIN dataset contains 10000 sentence pairs collected from Google News and split into training and test sets with an equal number of European Portuguese and

Brazilian Portuguese examples in each set. In my experiments, I will tackle only European Portuguese.

Training and test sets are imbalanced, thus a proportion of the majority class was taken as a baseline. Table 5.1 provides information about the training and test sets used.

	Number of examples	*Proportion of majority class (‘unrelated’)*
Training set	3000	68.2%
Test set	2000	69.3%

Table 5.1 Training and test set features.

The initial idea was to evaluate the model in which, instead of two questions, two sentences would be given as an input. I evaluated the model using different hyper-parameters, with and without word embeddings.

For preprocessing, I used NLTK Tokenizer (Bird et al., 2009) to split the sentences into tokens. All punctuation marks were dropped and all the tokens were lowercased.

Table 5.2 shows the results for different runs over the ASSIN test set.

	Test accuracy
BASELINE	69.3
Run 1	68.5
Run 2	69.1

Table 5.2 Evaluation results over the ASSIN test set.

Run 1 In this run I took exactly the same hyper-parameters as the ones used by Bogdanova et al. (2015). In Table 5.4, they are provided as reference. After 5 epochs of training, the model reaches more than 90% accuracy over the training data and begins to overfit, thus I stop training after the 5th epoch and evaluate the model using its final state. I used word embeddings for Portuguese from Rodrigues et al. (2016)[1] as initial values of a word representation layer of the CNN.

Run 2 For this run I changed the size of the convolutional layer from 300 to 1000; and, instead of using pre-trained word embeddings for word representation layer initial values, the values were randomly initialized using uniform distribution.

[1]https://github.com/nlx-group/lx-dsemvectors

Parameter	*Value*	*Description*
k	3	Size of k-gram
cl_u	300	Size of convolutional filter
batch size	1	Number of examples per gradient update
d	400	Size of word representation
epochs	5	Number of training epochs
word embeddings	Pre-trained	

Table 5.3 Run 1 hyper-parameters. First three lines indicate parameters whose values were identical to the model adopted in question similarity task. Last two lines indicate parameters that were set differently.

Parameter	*Value*	*Description*
cl_u	1000	Size of convolutional filter
word embeddings	Random (uniform)	

Table 5.4 Run 2 hyper-parameters. Only differences to Run 1 are shown.

The evaluation results for both runs are on a level with the baseline which shows that the model is applicable for the task of entailment detection for Portuguese.

These results concerning Portuguese entailment recognition are lower than the results concerning English question similarity because the training data set for the former is 8 times smaller than the one for latter. Enriching the training set might help to achieve higher accuracy for the task with Portuguese.

The evaluation results were not compared against another participants of the shared task for the reason that participants' results considered only the case of three-class classification.

5.3 Future work

The hyper-parameters were taken ad hoc and the accuracy can be further improved by tuning hyper-parameters automatically. A part of training data can be taken apart for validation and in order to pick better hyper-parameters the model can be tested over this validation set. Also, this method can be used for picking the best epoch for testing instead of taking the result after the 5th epoch.

Furthermore, the experiments should contain the extension of the model to the task of multi-class classification, namely three-class classification (entailment, paraphrase, unrelated) to make it comparable with other participants' result in the shared task.

5.4 Summary

This chapter has discussed the application of the convolutional neural network model I developed to the task of entailment recognition for Portuguese. Evaluation of this model has been shown that it is able to achieve an accuracy on a level with the baseline.

I have concluded the chapter with an overview of the future work that can be done in order to achieve better performance and obtain more comparable results.

The next chapter will present another application of the model, viz. concerned with paraphrase detection for Russian.

Chapter 6

Russian Paraphrase Detection Task

This chapter reports on the results of an experimental study on the application of convolution neural network to the shared task of sentence paraphrase detection for Russian. The Russian language contains richer morphology than English and Portuguese language, that were tackled in Chapters 4 and 5 respectively. Accordingly, tackling the Russian language required several improvements that are described in the chapter.

In Section 6.1, I will start with the task definition, which contains two runs. Then in Section 6.2, I will present the evaluation metrics and the results for both runs. Also, I will present two improvements that I brought into the convolutional neural network model, namely (1) the usage of several convolutional filters and (2) the usage of character embeddings instead of word embeddings. These improvements helped the model to obtain a competitive performance in the shared task. Section 6.3 will discuss how the performance of the system may be further enhanced.

6.1 Task definition

As indicated in the previous chapter, *paraphrases* are sentences in a mutual entailment relation, thus being semantically equivalent. Shared tasks which tackle the paraphrase identification, mostly for English, were organized under SemEval workshops (see Section 3.1.3 for further details).

The Russian language is a morphologically rich language with free word order and can be an interesting workbench for testing different models of paraphrase detection.

A core dataset for the task is ParaPhraser (Pronoza et al., 2015), a freely available corpus of Russian sentence pairs manually annotated as precise paraphrases, near-paraphrases and non-paraphrases. Each candidate pair was collected from news headlines and then manually annotated by three native speakers. Examples (1-3) show respective pairs of precise paraphrases, near-paraphrases and non-paraphrases taken from the corpus.

(1) A. КНДР аннулировала договор о ненападении с Южной Кореей.
DPRK annulled the non-aggression treaty with South Korea.
B. КНДР вышла из соглашений о ненападении с Южной Кореей.
DPRK withdrew from the non-aggression agreement with South Korea.

(2) A. ВТБ может продать долю в Tele2 в ближайшие недели.
VTB might sell its shares in TELE2 in the upcoming weeks.
B. ВТБ анонсировал продажу Tele2.
VTB announced the sale of TELE2.

(2) A. В главном здании МГУ загорелась столовая.
The student canteen was put on fire in the main building of MSU.
B. Из главного здания МГУ эвакуированы около 300 человек.
About 300 people are evacuated from the main building of MSU.

The size of the training set is 7000 pairs and the test set contains 1924 pairs.

The shared task consists of two tasks: one for three-class classification, and another for binary classification. I will tackle only the second task (Task 2) which is defined as following:

Given a pair of sentences, to predict whether they are paraphrases (whether precise or near paraphrases) or non-paraphrases.

There were two types of submissions: *standard run* which allowed only using the ParaPhraser corpus for training, and *non-standard run* which allowed using any other corpora. I participated in both types of runs.

6.2 Evaluation and results

6.2.1 Evaluation metrics

The results were accessed using two metrics: accuracy and F1 score. F1 is a harmonic mean between precision and recall:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6.1}$$

$$\text{precision} = \frac{tp}{tp + fp} \tag{6.2}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{6.3}$$

where:

tp , true positives, is the number of correctly classified paraphrases,

tn , true negatives, is the number of correctly classified non-paraphrases,

fp , false positives, is the number of pairs incorrectly classified as paraphrases,

fn , false negatives, is the number of pairs incorrectly classified as non-paraphrases.

6.2.2 Non-standard run

For the non-standard run I will describe the best result that I obtained. In the non-standard run it was permitted to use resources other than ParaPhraser corpus itself, as for example, word embeddings.

The word embeddings were taken from RusVectōrēs model (Kutuzov and Andreev, 2015) (CC Attribution 4.0 International license) trained on Russian national corpus.[1] All the words in the corpus were part-of-speech (PoS) tagged, lemmatised and lowercased.

In order to preprocess the input sentences, they were also lowercased, lemmatised and PoS-tagged using MyStem (Segalovich, 2003), the same tool that was reported by Kutuzov and Andreev (2015).

In order to speed-up the training of the network, I used Keras's RMSProp optimizer instead of using stochastic gradient descent (SGD). RMSProp is reported[2] to achieve high accuracy in few epochs. Below I provide other hyper-parameters that were used to obtain the best result for the non-standard run.

Furthermore, instead of combining representations of k-grams after the convolution by summing them up element-wise, I used a maximum function that picks element-wise maximum between all k-gram representations $\mathbf{z}_n$ (see Section 4.1.3 to recall how pooling layer operates).

[1]http://www.ruscorpora.ru

[2]https://blog.keras.io/category/tutorials.html

Parameter	*Value*	*Description*
k	3	Size of k-gram
cl_u	300	Size of convolutional filter
d	300	Size of word representation
epochs	5	Number of training epochs
pooling	MAX	pooling layer function
optimizer	RMSProp	Keras's optimizer
word embeddings	Pre-trained: RusVectōrēs	

Table 6.1 Non-standard run hyper-parameters.

	Accuracy	*F1*
BEST (True Positive)	77.39	81.10
NLX	69.94	76.80
BASELINE	49.66	54.03

Table 6.2 Results for non-standard run for task 2. My model is indicated as NLX.

Table 6.2 shows evaluation results together with the most frequent class baseline and the result of the best team that participated in the shared task.

6.2.3 Standard run

For the standard run, hereby I present two versions. The first one uses randomized *word embeddings* as the first layer of the CNN and the second run uses *character embeddings*. In both cases, as no external resources were allowed, tokens were only lowercased and punctuation was removed.

Furthermore, I used a more advanced method of convolution, that uses several convolutional filters with different sizes of k-gram. Figure 6.1 shows how this type of layer operates for $k \in \{2, 3\}$.

Word embeddings For this run, the input sentences were split into word tokens and each token was given an initial random uniform representation. Also, the idea of convolution was extended further.

Table 6.3 shows hyper-parameters that were used for this run.

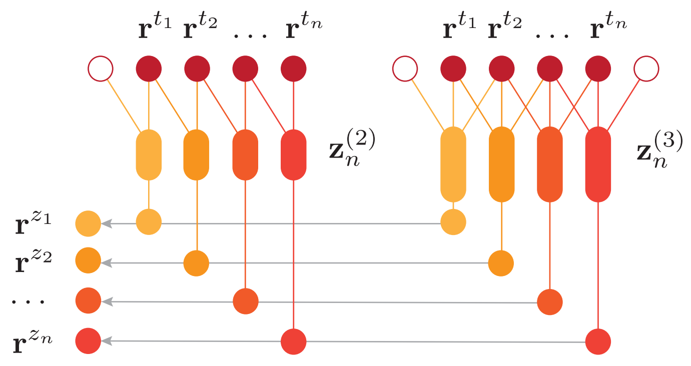
$\mathbf{r}^{t_1}$ $\mathbf{r}^{t_2}$ ... $\mathbf{r}^{t_n}$
$\mathbf{r}^{t_1}$ $\mathbf{r}^{t_2}$ ... $\mathbf{r}^{t_n}$
$\mathbf{z}_n^{(2)}$
$\mathbf{z}_n^{(3)}$
$\mathbf{r}^{z_1}$
$\mathbf{r}^{z_2}$
...
$\mathbf{r}^{z_n}$

Fig. 6.1 Concatenation of various convolution filters (for $k = 2$ and $k = 3$). Token (word/character) representation are denoted as $\mathbf{r}_{t_n}$. Concatenations of $k \in \{2, 3\}$ tokens are denoted as $\mathbf{z}_n^{(2)}$ and $\mathbf{z}_n^{(2)}$ respectively. Then activation function was applied to each of the concatenation and results were joined together into representations $\mathbf{r}^{z_i}$.

Parameter	*Value*	*Description*
k	$\{3, 5, 8, 12\}$	Sizes of k-grams
cl_u	100	Size of each convolutional filter
d	300	Size of word representation
epochs	5	Number of training epochs
pooling	MAX	pooling layer function
optimizer	RMSProp	Keras's optimizer

Table 6.3 Standard run hyper-parameters for NLX run, with word embeddings.

Parameter	*Value*	*Description*
k	$\{2, 3, 5, 7, 9, 11\}$	Sizes of k-grams
cl_u	100	Size of each convolutional filter
d	100	Size of character representation
epochs	20	Number of training epochs
pooling	MAX	pooling layer function
optimizer	RMSProp	Keras's optimizer

Table 6.4 Standard run hyper-parameters for NLX run, with character embeddings.

	Accuracy	*F1*
BEST (dups)	74.59	80.44
NLX (character embeddings)	72.74	78.80
NLX (word embeddings)	66.19	76.44
BASELINE	49.66	54.03

Table 6.5 Results for the standard run for task 2.

Character embeddings For this run, instead of tokenising sentences to words, I split sentences into characters. The main reason is that character-level embeddings are reported to be good in capturing morphological information (dos Santos and Gatti, 2014; Kim et al., 2016), that is important for such morphologically rich languages as Russian. Table 6.4 shows hyper-parameters that were used for the run.

Table 6.5 shows evaluation results, as reported by the Paraphraser.ru platform[3] for both runs together with the baseline and the result of the best team that participated in the shared task.

6.2.4 Evaluation upon the released test set

After the shared task was finished, organizers have published the test set together with correct labels. I tested my models locally upon it and obtained the results that vary due to the random initialization of the weights of the neural network.

Table 6.6 shows the results that were obtained locally compared to the results that were reported by the organizers of the shared task.

[3]http://www.paraphraser.ru/contests/result/?contest_id=1

	Local test	*Paraphraser.ru*
Non-standard run	68.71	69.94
Standard run (words)	65.18	66.19
Standard run (characters)	71.99	72.74

Table 6.6 Accuracy scores for Russian paraphrase detection obtained by submitting the output to the shared task organization (third column) and by running the output locally with the annotated test set (second column), with the same number of training epochs in both cases.

	Best epoch	*Paraphraser.ru*
Non-standard run	70.63	69.94
Standard run (words)	66.63	66.19
Standard run (characters)	73.90	72.74

Table 6.7 Accuracy scores as in the previous Table 6.6 with the difference that for the scores obtained locally, the model was trained with the best number of training epochs.

In the shared task, I submitted the result obtained by the neural network that was trained through all the epochs, whose number was indicated in Sections 6.2.2 and 6.2.3.

Nevertheless having access to the annotated test set, it is possible to calculate the best number of training epochs, which will lead to better results. Table 6.7 shows the resulting accuracy of the model if the best epoch is chosen and the next section below will suggest the possible way of choosing the right epoch.

6.3 Discussion and future work

The evaluation results for both runs are above the baseline and this shows that the model is applicable for the task of paraphrase detection for Russian.

The result for standard run is competing with the best system and can be further improved by tuning hyper-parameters automatically and also picking the epoch for testing automatically, based on the cross-validation results.

Surprisingly, the results for the standard run outperformed the ones for the non-standard run, though the non-standard run used external resources for lemmatisation and initial word embeddings. I assume that this is mainly due to higher focus of this work on standard run in conditions of time constraints of the competitive shared task. Results for the non-standard run could be further improved through choosing different word embedding models and by tuning the hyper-parameters.

6.4 Summary

This chapter has presented the results of the application of the convolutional neural network model to the task of paraphrase detection for Russian. Starting with the definition of the task, I discussed two improvements that changed the neural network architecture and helped to achieve competitive results in the task. The Russian language contains richer morphology than English and Portuguese. The improved model achieves competitive performance for the task of detecting if two sentences are paraphrases without using of any external resources.

Chapter 7

Conclusions

7.1 Summary of contributions

This dissertation was concerned with the task related to modelling semantic relations between the elements of pairs of texts.

Replicability In one of the experiments undertaken, the task was to compare two texts in order to draw a decision if they are semantically equivalent questions in English. I have implemented a convolutional neural network (CNN) introduced by Bogdanova et al. (2015) to assess the replicability of their strong results. I would like to indicate two main contributions related to this task:

- The first contribution shows that I could obtain comparable or better results with the same architecture and parameters of that CNN.
- As a second contribution, I have indicated that the work done by Bogdanova et al. (2015) presumably includes a drawback regarding text preprocessing which excludes removal of certain clues. These clues can strongly guide the system in detecting semantically equivalent questions in illegitimate way. Taking away the clues leads to significant drop in system performance, thus bringing performance scores to the range of state of the art.
- A third contribution was to improve Bogdanova et al. (2015) result, by taking initial random word embeddings, rather than pre-trained ones.

Task adaptation Another goal of my work was to see how the convolutional neural network models can help with another tasks concerned with the modelling of two sentences. I chosen two different tasks and two different languages: entailment recognition for Portuguese and paraphrase detection for Russian. The main contributions are the following:

- For the Portuguese task, I have shown that the given model can achieve the accuracy on a level with the baseline.
- For the Russian task, I have built a system that integrates character embeddings instead of word embeddings. This system does not use any external resources and achieves competitive performance for the task of detecting if two sentences are paraphrases.

A contribution that was common for all the tasks questioned the impact pre-trained of word embeddings for the specific task. I have shown that using of randomly initialised word embeddings instead of ones pre-trained with recurrent neural network improves the performance of the system regarding each of the tasks.

7.2 Further work

Regarding the question answering task, there is a variety of possible avenues of future research. The most obvious one would be to use slightly different and more widely used datasets such as the SemEval community question answering tasks' dataset (Agirre et al., 2016). This dataset, instead of labelling pairs as being duplicate or non-duplicate ranks the answers to the questions by the level of relatedness.

More deep and sophisticated architectures such as attention-based convolutional neural network (ABCNN), together with the use of additional textual features (Yin et al., 2016), can be resorted to possibly achieve a competitive performance for this task.

A very important aspect that may be further developed in orthogonal to the deep learning architecture that is used. This aspect is concerned with the tuning the hyper-parameters of neural network. The software that was developed might be further improved to make use of the validation set in order to automatically optimize the hyper-parameters. Different techniques such as grid search, random search and model-based optimizations can be experimented with (Goodfellow et al., 2016).

References

Agirre, E., Gonzalez-Agirre, A., Lopez-Gazpio, I., Maritxalar, M., Rigau, G., and Uria, L. (2016). Semeval-2016 task 2: Interpretable semantic textual similarity. In Bethard et al. (2016), pages 512–524.

Al-Rfou, R., Alain, G., Almahairi, A., Angermüller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J. B., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., de Brébisson, A., Breuleux, O., Carrier, P. L., Cho, K., Chorowski, J., Christiano, P., Cooijmans, T., Côté, M., Côté, M., Courville, A. C., Dauphin, Y. N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S., Dinh, L., Ducoffe, M., Dumoulin, V., Kahou, S. E., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I. J., Graham, M., Gülçehre, Ç., Hamel, P., Harlouchet, I., Heng, J., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrançois, S., Lemieux, S., Léonard, N., Lin, Z., Livezey, J. A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P., Mastropietro, O., McGibbon, R., Memisevic, R., van Merriënboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C. J., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlüter, J., Schulman, J., Schwartz, G., Serban, I. V., Serdyuk, D., Shabanian, S., Simon, É., Spieckermann, S., Subramanyam, S. R., Sygnowski, J., Tanguay, J., van Tulder, G., Turian, J. P., Urban, S., Vincent, P., Visin, F., de Vries, H., Warde-Farley, D., Webb, D. J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S., and Zhang, Y. (2016). Theano: A python framework for fast computation of mathematical expressions. *CoRR*, abs/1605.02688.

Alsulami, B. S., Abulkhair, M. F., and Eassa, F. E. (2012). Near duplicate document detection survey. *International Journal of Computer Science and Communications Networks*, 2(2):147–151.

Baroni, M., Bernardi, R., and Zamparelli, R. (2014). Frege in space: A program of compositional distributional semantics. *LiLT (Linguistic Issues in Language Technology)*, 9.

Barsalou, L. W. (2010). Grounded cognition: Past, present, and future. *Topics in Cognitive Science*, 2(4):716–724.

Bethard, S., Cer, D. M., Carpuat, M., Jurgens, D., Nakov, P., and Zesch, T., editors (2016). *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016.* The Association for Computer Linguistics.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly.

Bloom, P. (2001). *Précis of How children learn the meanings of words*, volume 24. Cambridge University Press.

Bogdanova, D., dos Santos, C. N., Barbosa, L., and Zadrozny, B. (2015). Detecting semantically equivalent questions in online user forums. In Alishahi, A. and Moschitti, A., editors, *Proceedings of the 19th Conference on Computational Natural Language Learning, CoNLL 2015, Beijing, China, July 30-31, 2015*, pages 123–131. ACL.

Branco, A., Rodrigues, L., Silva, J. R., and Silveira, S. (2008). Xisquê: An online QA service for portuguese. In Teixeira, A. J. S., de Lima, V. L. S., de Oliveira, L. C., and Quaresma, P., editors, *Computational Processing of the Portuguese Language, 8th International Conference, PROPOR 2008, Aveiro, Portugal, September 8-10, 2008, Proceedings*, volume 5190 of *Lecture Notes in Computer Science*, pages 232–235. Springer.

Bullinaria, J. A. and Levy, J. P. (2012). Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior Research Methods*, 44(3):890–907.

Cheng, J. and Kartsaklis, D. (2015). Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In Màrquez et al. (2015), pages 1531–1542.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

Collins, A. M. and Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Memory and Language*, 8(2):240.

Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*.

dos Santos, C. N. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In Hajic, J. and Tsujii, J., editors, *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 69–78. ACL.

dos Santos, C. N. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1818–1826. JMLR.org.

Erk, K. (2012). Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Ferrucci, D. A. (2012). Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3):1.

Fonseca, E. R., dos Santos, L. B., Criscuolo, M., and Aluísio, S. M. (2016). Assin: Avaliação de similaridade semântica e inferência textual. http://propor2016.di.fc.ul.pt/wp-content/uploads/2015/10/assin-overview.pdf.

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

Grefenstette, E. and Sadrzadeh, M. (2011). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1394–1404. ACL.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Haxby, J. V., Gobbini, M. I., Furey, M. L., Ishai, A., Schouten, J. L., and Pietrini, P. (2001). Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539):2425–2430.

He, H., Gimpel, K., and Lin, J. J. (2015). Multi-perspective sentence similarity modeling with convolutional neural networks. In Màrquez et al. (2015), pages 1576–1586.

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

Ji, Y. and Eisenstein, J. (2013). Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 891–896. ACL.

Jin, X., Zhou, Z., Lee, M. K. O., and Cheung, C. M. K. (2013). Why users keep answering questions in online question answering communities: A theoretical and empirical investigation. *Int J. Information Management*, 33(1):93–104.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In Schuurmans, D. and Wellman, M. P., editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2741–2749. AAAI Press.

Kutuzov, A. and Andreev, I. (2015). Texts in, meaning out: Neural language models in semantic similarity tasks for russian. volume 2, pages 133–144.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, pages 143–155.

Madnani, N., Tetreault, J. R., and Chodorow, M. (2012). Re-examining machine translation metrics for paraphrase identification. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 3-8, 2012, Montréal, Canada*, pages 182–190. The Association for Computational Linguistics.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60. The Association for Computer Linguistics.

Markoff, J. (2011). Computer wins on 'jeopardy!': trivial, it's not. *New York Times*, 16.

Màrquez, L., Callison-Burch, C., Su, J., Pighin, D., and Marton, Y., editors (2015). *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. The Association for Computational Linguistics.

McDonald, S. and Brew, C. (2004). A distributional model of semantic context effects in lexical processing. In Scott, D., Daelemans, W., and Walker, M. A., editors, *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 17–24. ACL.

Mikolov, T., Yih, W., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In Vanderwende, L., III, H. D., and Kirchhoff, K., editors, *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751. The Association for Computational Linguistics.

Mitchell, T. M. (1997). *Machine learning.* McGraw Hill series in computer science. McGraw-Hill.

Mitkov, R. (2005). *The Oxford handbook of computational linguistics.* Oxford University Press.

Murphy, B., Talukdar, P. P., and Mitchell, T. M. (2012). Selecting corpus-semantic models for neurolinguistic decoding. In Agirre, E., Bos, J., and Diab, M. T., editors, *Proceedings of the First Joint Conference on Lexical and Computational Semantics, *SEM 2012, June 7-8, 2012, Montréal, Canada.*, pages 114–123. Association for Computational Linguistics.

Nakov, P., Màrquez, L., Moschitti, A., Magdy, W., Mubarak, H., Freihat, A. A., Glass, J., and Randeree, B. (2016). Semeval-2016 task 3: Community question answering. In Bethard et al. (2016), pages 525–545.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Pronoza, E. V., Yagunova, E., and Pronoza, A. (2015). Construction of a Russian paraphrase corpus: Unsupervised paraphrase extraction. In Braslavski, P., Markov, I., Pardalos, P. M., Volkovich, Y., Ignatov, D. I., Koltsov, S., and Koltsova, O., editors, *Information Retrieval - 9th Russian Summer School, RuSSIR 2015, Saint Petersburg, Russia, August 24-28, 2015, Revised Selected Papers*, volume 573 of *Communications in Computer and Information Science*, pages 146–157. Springer.

Rapp, R. (2004). A freely available automatically generated thesaurus of related words. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association.

Rodrigues, J. A., Branco, A., Neale, S., and Silva, J. R. (2016). Lx-dsemvectors: Distributional semantics models for portuguese. In Silva, J. R., Ribeiro, R., Quaresma, P., Adami, A., and Branco, A., editors, *Computational Processing of the Portuguese Language - 12th International Conference, PROPOR 2016, Tomar, Portugal, July 13-15, 2016, Proceedings*, volume 9727 of *Lecture Notes in Computer Science*, pages 259–270. Springer.

Rubenstein, H. and Goodenough, J. B. (1965). Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Searle, J. R. (1988). Minds, brains, and programs. *Readings in Cognitive Science*, page 20–31.

Segalovich, I. (2003). A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. In Arabnia, H. R. and Kozerenko, E. B., editors, *Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications. MLMTA'03, June 23 - 26, 2003, Las Vegas, Nevada, USA*, pages 273–280. CSREA Press.

Smith, E. E., Shoben, E. J., and Rips, L. J. (1974). Structure and process in semantic memory: A featural model for semantic decisions. *Psychological Review*, 81(3):214–241.

Sultan, M. A., Bethard, S., and Sumner, T. (2015). Dls@cu: Sentence similarity from word alignment and semantic vector composition. In Cer, D. M., Jurgens, D., Nakov, P., and Zesch, T., editors, *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 148–153. The Association for Computer Linguistics.

Von Melchner, L., Pallas, S. L., and Sur, M. (2000). Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404(6780):871–876.

Wittgenstein, L. (1953). *Philosophical Investigations*. Basil Blackwell, Oxford.

Woods, W. A. and Kaplan, R. (1977). Lunar rocks in natural english: Explorations in natural language question answering. *Linguistic Structures Processing*, 5:521–569.

Wu, Y., Zhang, Q., and Huang, X. (2011). Efficient near-duplicate detection for q&a forum. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 1001–1009. The Association for Computer Linguistics.

Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016). ABCNN: attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics (TACL)*, 4:259–272.

Appendix A

User guide

HTML version of this user guide is available on a server of NLX-Group:
http://nlx-server.di.fc.ul.pt/dissertations/Maraev/README.html
(username and password are provided upon request).

A.1 Prerequisites

1. Download the source code:
   http://nlx-server.di.fc.ul.pt/dissertations/Maraev/msrdsdl.tar.gz
   (username and password are provided upon request).
2. Extract the code:
   ```
   tar -xvf msrdsl.tar.gz
   cd msrdsdl
   ```
3. You will need **Python version 3.4.3 or higher**.
4. Install required packages:
   ```
   pip install -r requirements.txt
   ```
5. Set up Theano backend for Keras by editing the configuration file `~/.keras/keras.json` and changing the field `backend` to `"theano"`.
6. If you want to run experiments that require word embeddings you need to download and extract distributional models:
   ```
   wget http://nlx-server.di.fc.ul.pt/dissertations/Maraev/models.tar.gz
   tar -xvf models.tar.gz
   ```

A.2 About the program

After satisfying all the prerequisites you will have the following directory structure:

```
|-- cnn.py
|-- data
|   |-- askubuntu
|   |   |-- clue
|   |   |   |-- test.tsv
|   |   |   |-- train.tsv
|   |   |   '-- val.tsv
|   |   '-- noclue
|   |       |-- test.tsv
|   |       |-- train.tsv
|   |       '-- val.tsv
|   |-- meta
|   |   |-- clue
|   |   |   |-- test.tsv
|   |   |   |-- train.tsv
|   |   |   '-- val.tsv
|   |   '-- noclue
|   |       |-- test.tsv
|   |       |-- train.tsv
|   |       '-- val.tsv
|   |-- pt
|   |   |-- test.tsv
|   |   |-- train.tsv
|   '-- ru
|       |-- test.tsv
|       '-- train.tsv
|-- __init.py__
|-- models
|   |-- askubuntu.w2v
|   |-- meta.w2v
|   |-- pt.w2v
|   '-- ruscorpora.model.w2v
|-- preprocess.py
|-- README.org
'-- requirements.txt
```

You can run the application `./cnn.py` with the `--help` argument to see available parameters.

To change hyperparameters you can modify the method `SentenceSimilarity.set_hyperparameters` by adding new modes.

A.3 Replication of my work

A.3.1 Question Answering

Replication of the work by Bogdanova et al. (2015)

For AskUbuntu dataset:

```
./cnn.py replication --train data/askubuntu/clue/train.tsv \
         --val data/askubuntu/clue/val.tsv \
         --test data/askubuntu/clue/test.tsv \
         --w2v models/askubuntu.w2v
```

For META Stackexchange dataset:

```
./cnn.py replication --train data/meta/clue/train.tsv \
         --val data/meta/clue/val.tsv \
         --test data/meta/clue/test.tsv \
         --w2v models/meta.w2v
```

Impact of text preprocessing (clue phrases removed)

For AskUbuntu dataset:

```
./cnn.py pp_impact --train data/askubuntu/noclue/train.tsv \
         --val data/askubuntu/noclue/val.tsv \
         --test data/askubuntu/noclue/test.tsv \
         --w2v models/askubuntu.w2v
```

For META Stackexchange dataset:

```
./cnn.py pp_impact --train data/meta/noclue/train.tsv \
         --val data/meta/noclue/val.tsv \
         --test data/meta/noclue/test.tsv \
         --w2v models/meta.w2v
```

Impact of word embeddings (no pre-trained word embeddings)

```
./cnn.py we_impact --train data/askubuntu/noclue/train.tsv \
         --val data/askubuntu/noclue/val.tsv \
         --test data/askubuntu/noclue/test.tsv
```

A.3.2 Portuguese Entailment Recognition

Run 1

```
./cnn.py pt_1 --train data/pt/train.tsv \
         --val data/pt/test.tsv \
         --w2v models/pt.w2v
```

Run 2

```
./cnn.py pt_2 --train data/pt/train.tsv --val data/pt/test.tsv
```

A.3.3 Russian Paraphrase Detection

Non-standard run

```
./cnn.py ru_ns --train data/ru/train.tsv \
         --val data/ru/test.tsv \
         --w2v models/ruscorpora.model.w2v
```

Standard run

1. Word embeddings

   ```
   ./cnn.py ru_word --train data/ru/train.tsv --val data/ru/test.tsv
   ```

2. Character embeddings

   ```
   ./cnn.py ru_char --train data/ru/train.tsv --val data/ru/test.tsv
   ```