Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 2086–2094
Marseille, 11–16 May 2020

Automated Discovery of Mathematical Definitions in Text

Natalia Vanetik, Marina Litvak, Sergey Shevchuk and Lior Reznik
Department of Software Engineering
Shamoon College of Engineering, Beer Sheva, Israel
{natalyav,marinal,sergesh}@sce.ac.il, liorrezn@gmail.com

Abstract

Automatic definition extraction from texts is an important task that has numerous applications in several natural language processing fields such as summarization, analysis of scientific texts, automatic taxonomy generation, ontology generation, concept identification, and question answering. For definitions that are contained within a single sentence, this problem can be viewed as a binary classification of sentences into definitions and non-definitions. Definitions in scientific literature can be generic (Wikipedia) or more formal (mathematical articles). In this paper, we focus on automatic detection of one-sentence definitions in mathematical texts, which are difficult to separate from surrounding text. We experiment with several data representations, which include sentence syntactic structure and word embeddings, and apply deep learning methods such as convolutional neural network (CNN) and recurrent neural network (RNN), in order to identify mathematical definitions. Our experiments demonstrate the superiority of CNN and its combination with RNN, applied on the syntactically-enriched input representation. We also present a new dataset for definition extraction from mathematical texts. We demonstrate that the use of this dataset for training learning models improves the quality of definition extraction when these models are then used for other definition datasets. Our experiments with different domains approve that mathematical definitions require special treatment, and that using cross-domain learning is inefficient.

Keywords: definition extraction, deep learning

1. Introduction

Definitions have a very important role in scientific literature because they define the major concepts an article operates with. They are basic building blocks of a scientific article that are used to properly describe hypotheses, experiments and analyses. Definitions are used in many automatic text analysis tasks, such as question answering, ontology matching and construction, formal concept analysis and structural text summarization. Therefore, automatic definition extraction (DE) is an important field in natural language processing and can be used for better text analysis and search.

Definitions play a key role in mathematics, but their creation and use differs from those of "everyday language" definitions (a comprehensive study is given in series of works by Edwards and Ward in (Edwards and Ward, 2008; Edwards and Ward, 2004; Edwards, 1998) inspired by writings of Richard Robinson (Robinson, 1962) and lexicographer Sidney Landau (Landau, 2001)). They distinguish between extracted definitions report usage and have a truth value (Edwards and Ward, 2004), while stipulated definitions create usage, indeed create concepts, by decree but have no truth value. Mathematical definitions frequently do have a history as they evolve over time. The definition we use for function, for instance, may not be the one that was used a hundred years ago. The concept of connectivity has two definitions, path-connected and set-theoretically connected (Edwards and Ward, 2008). Van Dormolen and Zaslavsky (Van Dormolen and Zaslavsky, 2003) describe a good mathematical definition as containing criteria of *hierarchy*, *existence*, *equivalence*, and *acclimatization*. Desired but not necessary criteria of a definition are *minimality*, *elegance* and *degenerations*. Not every definition appearing in text is mathematical in the above sense. For example, Wikipedia articles contain definitions of different style. We see below two Wikipedia definitions, where just one of them is a mathematical definition.

Definition 1:
***Kane & Abel**, formally known as 'Double Vision', is an American hip hop duo formed by twin brothers Daniel and David Garcia that were founded by Master P in late 1995. They were best known for their time with No Limit Records.*

Definition 2:
*In abstract algebra, an **abelian group**, also called a **commutative group**, is a group in which the result of applying the group operation to two group elements does not depend on the order in which they are written. That is, these are the groups that obey the axiom of commutativity. Abelian groups generalize the arithmetic of addition of integers. They are named after early 19th century mathematician Niels Henrik Abel.*

Naturally, we expect to find formal mathematical definitions in highly abstract texts such as mathematical articles. The negative aspect of a mathematical definition formulation is the extensive use of formulas and notations, both in definitions and in surrounding text. The number of words in such sentences is smaller than expected due to formulas, and the sentences that do not contain definitions also use a lot of notations and formulas. As an example of such text, Definition 3 below contains highly formal definition from Wolfram MathWorld. The first sentence in this text is considered a definition sentence.

Definition 3:
*Also called Macaulay ring, a **Cohen Macaulay ring** is a Noetherian commutative unit ring R in which any proper ideal I of height n contains a sequence $x_1, \ldots, x_n$ of elements (called a ring regular sequence such that for all $i = 1, \ldots, n$, the residue class of x_i in the quotient ring $R/\langle x_1, \ldots, x_{i-1} \rangle$ is a non-zero divisor. If $x_1, \ldots, x_n$ are indeterminate over a field K, the above condition is fulfilled by the maximal ideal $I =< x_1, \ldots, x_n >$.*

Current methods for automatic DE view it as a binary classification task, where a sentence is classified as a definition or a non-definition. A supervised learning process is usually used for this task, employing feature engineering for sentence data. The absolute majority of current methods study generic definitions and not mathematical definitions (see Section 2.).

In this paper we describe a supervised learning method for automatic DE from mathematical texts. Our method applies convolutional neural network (CNN), recurrent neural network (RNN), and their combinations to the raw text data and sentence syntax structure, in order to detect definitions. Our method is evaluated on three different corpora; two are well-known corpora for generic DE and one is a new annotated corpus of mathematical definitions, introduced in this paper.

The main contributions of this paper are (1) a new annotated corpus of mathematical definitions, (2) an adaptation of deep neural networks for DE that uses new representations of input sentences fed to these networks, and (3) extensive experiments with multiple network and input configurations that are performed on different datasets. These all contribute to showing that using specifically suited training data along with suggested sentence representation significantly improves extraction of mathematical definitions.

The paper is organized as follows. Section 2. contains a survey of up-to-date related work. Section 3. describes datasets, preprocessing, and the structure of neural networks used in our approach. Section 4. provides evaluation results and their analysis. Section 5. contains our conclusions.

2. Related Work

Prior work in the field of DE can be divided into three main categories: (1) rule-based methods, (2) machine-learning methods relying on manual feature engineering, and (3) methods that use deep learning techniques.

Early works about DE from text documents belong to the first category. These works rely mainly on manually crafted rules based on linguistic parameters (Schuyler et al., 1993; Klavans and Muresan, 2001; Schuyler et al., 1993; Saggion and Gaizauskas, 2004; Storrer and Wellinghoff, 2006; Borg et al., 2009).

The second category of DE algorithms relies on semi-supervised and supervised machine learning that use semantic and other features to extract definitions. This approach generates DE rules automatically but relies on feature engineering to do so (Fahmi and Bouma, 2006; Westerhout et al., 2007; Westerhout, 2009; Navigli and Velardi, 2010). Many recent works explored lexico-syntactic and syntactic dependencies in order to distinguish between definitions and regular texts (Reiplinger et al., 2012; Boella and Di Caro, 2013; Espinosa-Anke and Saggion, 2014; Anke et al., 2015). The DefMiner system, proposed in (Jin et al., 2013), uses Conditional Random Fields (CRF) and hand-crafted shallow parsing patterns to identify definitions in both word and sentence levels.

Algorithms in the third category use Deep Learning (DL) techniques for DE, often incorporating syntactic features into the network structure. Li et al. (2016) uses Long Short-Term Memory (LSTM) and word vectors to identify definitions and then tests this approach on the English and Chinese texts. The authors of (Anke and Schockaert, 2018) combine CNN and RNN, based on syntactic features and word vector representation of sentences for DE. We use the approach of (Anke and Schockaert, 2018) as a starting point and as a baseline for our method. The difference between our method and the above algorithms is a novel text representation and network architectures that ensure better performance.

Our approach is based on the observation that the definition sentences are usually different from regular sentences in both lexical and grammatical levels. Therefore, we hypothesize that representing sentences with both types of information should improve their inference. We follow the standard approach to capture the semantic differences between texts by considering word embeddings instead words themselves. We also hypothesize that the automatic feature engineering performed by neural networks and context-aware classification models can assist each other and achieve better performance cooperatively than separately.

3. Methodology

Our approach uses a matrix representation of a sentence, where every word and every syntactic dependency in that sentence is represented by a word vector (Figure 1 depicts the pipeline).

We define several deep neural network architectures that combine CNN and RNN layers in a different way. We train every network on preprocessed[1] text data, where every sentence is labeled as a definition or a non-definition. During testing, we use the pre-trained network for DE.

3.1. Sentence representation parameters

To represent *sentence words*, we use standard sentence modeling for CNNs (Kim, 2014), where every word w is represented by its k-dimensional word vector $\vec{w}$ (Mikolov et al., 2013), and all sentences are assumed having the same length n, using zero padding where necessary. An entire sentence is then represented by $n \times k$ zero-padded matrix $S_{n\times k}$. We use $k = 300$ and $n = \max_i\{\text{length of sentence \#i}\}$.

Syntactic dependency, in the dependency parse tree of a sentence, has the form (w_i, w_j, d_{ij}), where w_i, w_j are words and d_{ij} is the dependency label.[2] For example, a tuple $(family, Amiga, nsubj)$ represents dependency named $nsubj$, which connects the word $family$ with word the $Amiga$.

We represent the *dependency words* w_i, w_j of a dependency (w_i, w_j, d_{ij}) by a single vector, denoted by $\vec{r_{ij}}$, computed in one of the following ways:

- Normalized sum $\vec{r_{ij}}^{avg} := \frac{1}{2}(\vec{w_i}+\vec{w_j})$ of word vectors $\vec{w_1}$ and $\vec{w_2}$. The resulting vector has 300 dimensions.
- Concatenation $\vec{r_{ij}}^{c} := \vec{w_1} \circ \vec{w_2}$ of the corresponding word vectors $\vec{w_1}$ and $\vec{w_1}$. The resulting vector has 600 dimensions.

[1] We applied the following text preprocessing steps: sentence boundary detection, tokenization, and dependency parsing with Stanford CoreNLP package (Manning et al., 2014).

[2] The Stanford CoreNLP parser supports 46 dependency types.

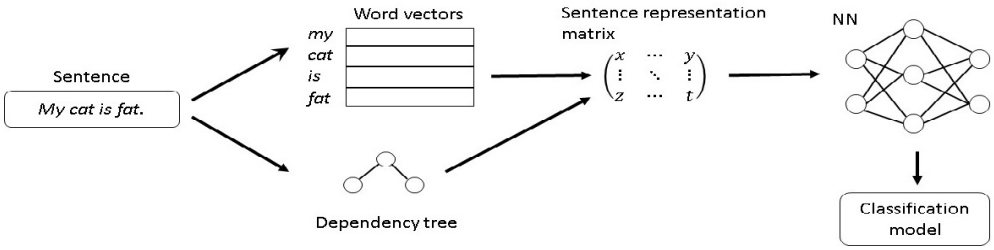

Figure 1: Pipeline of our approach

The dependency label d_{ij} is represented in one of the following ways:

- One-hot representation dep_{ij} of the dependency label over the search space of size 46.
- Concatenation $dep_{ij} \circ depth_{ij}$ of one-hot dependency label representation with the depth vector $depth_{ij}$ containing one number—the depth $n \in \mathbb{Z}^{\geq 0}$ of the dependency edge in the dependency tree (edges starting in the root of the tree have depth 0).

3.2. Neural network models

In this section we describe different neural network models we have implemented and tested in this work.

3.2.1. Neural network structures

We use several different network configurations, described below:

1. The convolutional network, denoted by CNN uses a convolutional layer (LeCun et al., 1998) only.[3]
2. The $CBLSTM$ network uses a CNN layer followed by a bidirectional LSTM layer, following the approach of (Anke and Schockaert, 2018).[4]
3. The recurrent network RNN that uses a single LSTM layer.
4. The $BLSTMCNN$ network that uses a bidirectional LSTM layer followed by a CNN layer.

The main idea behind the choice of network configurations was to examine the stacking orders of the CNN and RNN layers to determine which is more beneficial for the DE task.[5]

3.2.2. Input representation

The final sentence representation is a concatenation of the word vector matrix with the dependency structure representation, enriched with the dependency label information.[6] Below, we outline three main input configurations that we have defined and tested on all our networks.

1. Configuration m includes word vectors for sentence words and the words of dependencies (normalized sum of word vectors). Formally, $m = S_{n\times k} \circ [\vec{r_{ij}}^{avg}]_{ij}$
2. Configuration ml includes word vectors for sentence words, dependency words, and dependency label representations. Formally, $ml = S_{n\times k} \circ [\vec{r_{ij}}^{avg} \circ dep_{ij}]_{ij}$
3. Configuration mld has full dependency information, including concatenation of word vectors for dependency words, dependency label, and dependency depth. Formally, $mld = [\vec{r_{ij}}^{c} \circ dep_{ij} \circ depth_{ij}]_{ij}$

Figures 2 (A), 2 (B), and 2 (C) show how dependencies are represented for configurations m, ml, and mld, respectively.

4. Experiments

Tests were performed on an Asus-X556U PC with 16GB of RAM, 100 GB of PAGE memory, and an Intel Core I7-7500U 2.70 GHz CPU.

4.1. Tools

The models were implemented with help of the following tools: (1) Stanford CoreNLP wrapper (Manning et al.,) for Python (tokenization, sentence boundary detection, and dependency parsing) , (2) gensim (Řehůřek and Sojka, 2010) (loading word2vec vectors), (3) Keras (Chollet and others, 2015) with Tensorflow (Abadi et al., 2015) as a back-end (NN models), and (4) Scikit-Learn (Pedregosa et al., 2011) (evaluation with F1, recall, and precision metrics). All networks were trained with batch size 32 and 10 epochs.

4.2. Data

In our work we use the three datasets–WCL, W00 and WFM–that are described below. For in-domain tests, every dataset was evaluated on its own. In cross-domain tests, a network or a baseline was trained on one of the datasets and was tested on another. Additionally, we used joint multi-domain dataset that contains the mathematical WFM dataset with the other two datasets, denoted WCL&W00&WFM. The number of sentences for each class, total number of sentences, majority vote values, total number of words, and number of joint words with two used word embedding models —Word2Vec and FastText[7]—are given in Table 1.

[3] The convolutional layer applies a filter to each n-gram window of 3 tokens, implying 3-grams. As other parameters, the window size was set empirically.

[4] The CNN layer is followed by the max pooling layer with a size 4.

[5] Experiments with different number of CNN and LSTM layers showed no improvements on the experimental data.

[6] Zero-padding was used to ensure the same number of rows in concatenated matrices.

[7] The self-embedding model, trained on the experimental data itself, was also evaluated. However, due to a worse performance, its results are not reported here.

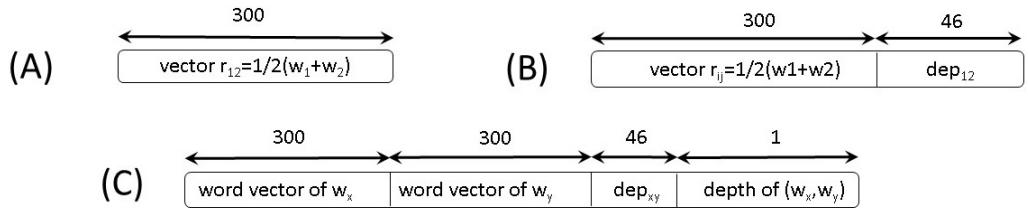

Figure 2: Input configurations

Dataset	Definition sentences	Non-definitions	Total sentences	Majority vote	Total words	Intersection with Word2Vec	Intersection with FastText
WCL	1871	2847	4718	0.603	21843	14368	16937
W00	731	1454	2185	0.665	7478	5307	6077
WFM	811	982	1793	0.548	4939	3379	3972
WCL&W00&WFM	3413	5283	8696	0.608	27822	17013	20590

Table 1: Datasets description

4.2.1. The WCL Dataset

The World-Class Lattices (WCL) dataset (Navigli et al., 2010), introduced in (Navigli et al., 2010), was constructed from manually annotated Wikipedia data in English. The version that we used (WCL v1.2) contains 4719 annotated sentences, 1,871 of which are proper definitions and 2,847 are *distractor* sentences, that have similar structures with proper definitions, but are not actually definitions. This dataset contains generic definitions in all areas and it is not mathematically oriented. A sample definition sentence from this dataset is

> *American Standard Code for Information Interchange (TARGET) is a character encoding based on English alphabet.*

and a sample distractor is

> *The premise of the program revolves around TARGET, Parker an 18-year-old country girl who moves back and forth between her country family, who lives on a bayou houseboat, and the wealthy Brents, who own a plantation and pancake business.*

In this corpus, following parts of definitions are annotated: (1) the DEFINIENDUM field (DF), referring to the word being defined and its modifiers, (2) the DEFINITOR field (VF), referring to the verb phrase used to introduce a definition, (3) the DEFINIENS field (GF) which includes the genus phrase, and (4) the REST field (RF), which indicates all additional sentence parts. According to the original annotations, existence of the first three parts indicates that a sentence is a definition.

4.2.2. The W00 Dataset

The W00 dataset (Jin et al., 2013) was compiled from ACL-ARC ontology (Bird et al., 2008) and contains 2185 manually annotated sentences, with 731 definitions and 1454 distractors; the style of the distractors is different from the one used in the WCL dataset. A sample definition sentence from this dataset is

> *Our system, SNS (pronounced "essence"), retrieves documents to an unrestricted user query and summarizes a subset of them as selected by the user.*

and a sample distractor is

> *The senses with the highest confidence scores are the senses that contribute the most to the function for the set.*

Annotation of the W00 dataset is token-based, with each token in a sentence identified by a single label that indicates whether a token is a part of a term (T), a definition (D), or neither (O). According to the original annotation, a sentence is considered to be a definition if every token in it is labeled D.

4.2.3. The WFM Dataset

The WFM dataset (Vanetik et al., 2019) was created by us by collecting and processing 2352 articles from Wolfram Mathworld (Weisstein and others, 2007). The dataset contains 1793 sentences, of which 811 are definitions and 982 are non-definitions. Sentences were extracted automatically and then manually separated into two categories: definitions and statements (non-definitions). It is freely available for download from GitHub. A sample statement sentence from this dataset is

> *The* $(7, 3, 2)$*-von Dyck group, also sometimes termed the* $(2, 3, 7)$*-von Dyck group, is defined as the von Dyck group with parameters* $(7, 3, 2)$.

and a sample non-definition is

> *Any* 2*-Engel group must be a group of nilpotency class* 3.

4.3. Text preprocessing

With regard to all three datasets described above, we applied the same text preprocessing steps in the following manner:

- Sentence splitting was derived explicitly from the datasets, without applying any additional procedure, in the following manner: WCL and W00 datasets came pre-splitted, and sentence splitting for the new WFM dataset was performed manually by our team.
- Tokenization and dependency parsing were performed on all sentences with the help of the Stanford CoreNLP

package (Manning et al., 2014). Because we filtered-out the special symbols (and therefore many formulas and notations) from a text, the sentences with less than 5 words were discarded from our analysis.

- For the WCL and W00 datasets used in (Anke and Schockaert, 2018) for DE, we replaced parsing by SpaCy (Honnibal and Johnson, 2015) with the Stanford CoreNLP parser. We found that the latter tool provided improved precision.

- We have tested two word embeddings options, the Word2Vec (Google, 2013) with pre-trained vectors obtained from the Google News corpus and fastText (Grave et al., 2018) with vectors pre-trained on English webcrawl and Wikipedia (available at (Grave et al., 2018)).

4.4. Baselines and SOA

Using the WEKA software, we applied the following baselines: Simple Logistic Regression (SLR), Support Vector Machine (SVM) and Random Forest (RF). To use these methods on complete sentences, we computed vector representation for every sentence as an average of word vectors for all its words, and marked sentences as belonging to either 'yes' (a definition) or 'no' (a non-definition) class. We have also used the DefMiner system of (Jin et al., 2013), available at `https://github.com/YipingNUS/DefMiner`, as a baseline. Because DefMiner comes pre-trained, we do not use it for the cross-domain evaluations.

As SOA methods, we applied CNN and LSTM networks on our representation models. Coming inline with (Anke and Schockaert, 2018), the best result among SOA methods was obtained by CNN on most datasets.

4.5. Results

We tested all network configurations and baselines (where applicable) in three domain configurations described in following sections. We use the notation $Network_{input}$ notation for a network of type $Network$, accepting input of type $input$ as specified in Section 3.2.2.

In total, we have 4 network types, 3 input configurations, and two word embedding models for each network, resulting in 24 final configurations. Our motivation was to test the extent to which the order and presence of CNN and RNN layers affect the results, and how the performance was affected by the representation of sentence dependency information and different embedding models. In the tables containing results, we show accuracy and F1 scores for both embeddings, with the best scores for each dataset marked in bold.

4.5.1. In-domain results

The first series of tests was performed for every dataset separately, using a random 66%-33% split of these datasets into test and training sets, respectively. Results are given in Table 2. Also, the chart in Figure 3 visualizes the performance of all tested models, in comparison with four baselines. Because DefMiner does not work with embedding vectors, its scores for both embeddings are the same. The first observation that can be seen from the chart on Figure 3 is that all NN-based models perform much better than four baselines. From Table 2 it can be seen that $\mathrm{CNN_{ml}}$ outperforms other models in most scores and datasets. Other models having best scores for some evaluation metrics are: $\mathrm{CBLSTM_{m}}$, $\mathrm{CNN_{mld}}$, $\mathrm{CBLSTM_{ml}}$, and $\mathrm{CBLSTM_{mld}}$. As such, the following conclusions can be made: (1) the CNN model—pure or integrated with BLSTM—achieves better performance than RNN; (2) the CNN model gains better performance with dependency information; (3) CNN layer followed by a bidirectional LSTM layer is the best combination of CNN and LSTM models for the DE task among tested combinations. The superiority of models having CNN layer as the first (or only) layer can be explained by the ability of CNN to learn features and reduce the number of free parameters in a high-dimensional sentence representation, allowing the network to be more accurate with fewer parameters.

4.5.2. Cross-domain results

The second series of tests was performed using one of the datasets as a training set, and another one as a test set, where WFM is either test or training dataset. The aim of these tests was to see how well mathematical definitions could be located using general datasets as a training set, and also whether training the system on mathematical definitions improved recognition of general definitions. Results are given in Table 3. Also, the chart in Figure 4 visualizes the performance of all tested models in cross-domain experiment, in comparison with three baselines. The outcomes that are observed in this experiment are very similar to what was observed in in-domain runs. The deep neural network models using three representations (see Section 3.2.2.), significantly outperformed all the baselines. From Table 3 it can be seen that $\mathrm{CBLSTM_{m}}$ and $\mathrm{CBLSTM_{ml}}$ outperform other models in most scores and datasets. Other models having best scores for some evaluation metrics are: $\mathrm{CNN_{ml}}$ and $\mathrm{CNN_{mld}}$. As such, we can get the same conclusions as for in-domain experiment: (1) the CNN model—pure or integrated with BLSTM—achieves better performance than RNN; (2) the CNN model usually gains better performance with dependency information; (3) CNN layer followed by a BLSTM layer is better combination for DE task than BLSTM followed by CNN, which can be explained by the ability of CNN to learn features and reduce the number of free parameters in a high-dimensional sentence representation.

Another important observation can be made if we compare between performance of in-domain and cross-domain experiments. The performance of all models in cross-domain learning is far below their performance in in-domain learning.[8] As such, we can conclude that all three datasets represent different domains, and using cross-domain learning with them is inefficient. As a private case, mathematical definitions require special treatment. Models trained to detect general definitions are not sufficient for the detection of mathematical definitions. Likewise, models trained on mathematical domain are not very helpful (below 80% of accuracy) for detection of general definitions.

[8] The best scores of cross-domain learning in non-mathematical domain (WCL<->W00) were limited by $0.70 - 0.73$.

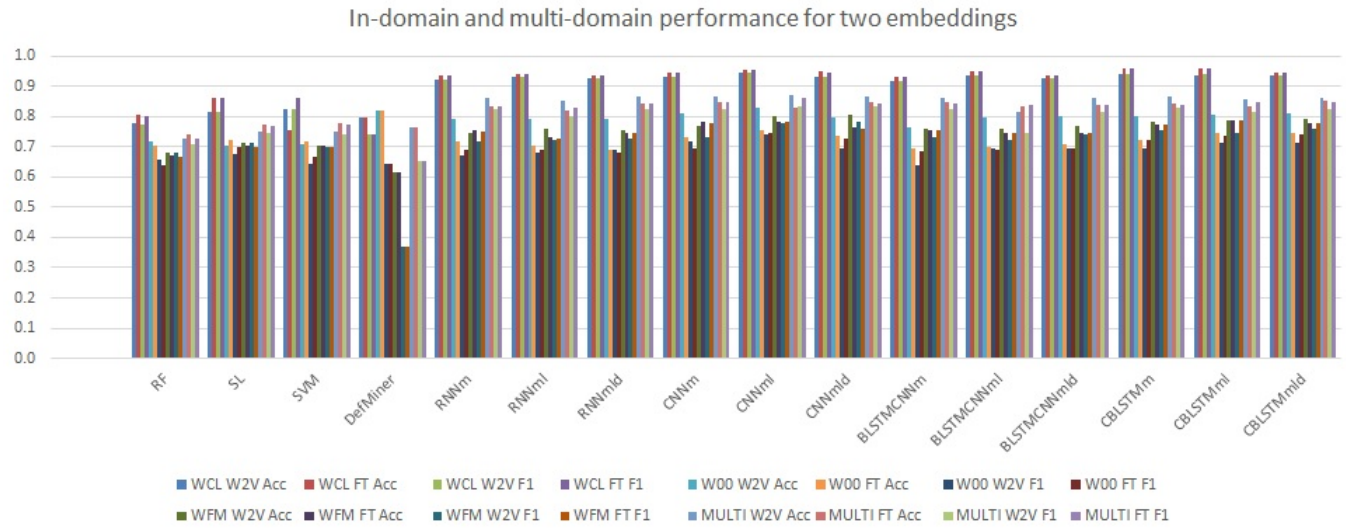

Figure 3: In-domain and multi-domain performance.

Dataset	WCL				W00				WFM				MULTI			
Method	W2V acc	FT acc	W2V F1	FT F1	W2V acc	FT acc	W2V F1	FT F1	W2V acc	FT acc	W2V F1	FT F1	W2V acc	FT acc	W2V F1	FT F1
RF	0.779	0.807	0.773	0.8	0.717	0.705	0.659	0.637	0.679	0.671	0.679	0.666	0.726	0.74	0.709	0.727
SL	0.817	0.859	0.817	0.859	0.703	0.721	0.677	0.697	0.714	0.703	0.714	0.699	0.750	0.773	0.743	0.770
SVM	0.824	0.756	0.824	0.863	0.707	0.717	0.645	0.665	0.704	0.704	0.701	0.698	0.752	0.779	0.741	0.775
DefMiner	0.797	0.797	0.741	0.741	0.819	0.819	0.644	0.644	0.617	0.617	0.371	0.371	0.766	0.766	0.654	0.654
$\mathrm{RNN_m}$	0.920	0.936	0.921	0.936	0.792	0.717	0.671	0.691	0.744	0.755	0.716	0.750	0.861	0.831	0.822	0.833
$\mathrm{RNN_{ml}}$	0.931	0.940	0.931	0.940	0.790	0.702	0.682	0.691	0.757	0.733	0.722	0.727	0.852	0.820	0.801	0.828
$\mathrm{RNN_{mld}}$	0.928	0.935	0.928	0.936	0.793	0.688	0.687	0.680	0.754	0.747	0.725	0.746	0.864	0.845	0.826	0.845
$\mathrm{CNN_m}$	0.933	0.947	0.933	0.947	0.811	0.731	0.715	0.694	0.767	0.780	0.732	0.778	0.867	0.847	0.826	0.846
$\mathrm{CNN_{ml}}$	**0.946**	0.956	**0.946**	0.956	**0.828**	**0.755**	**0.740**	**0.746**	0.800	0.784	0.778	0.783	**0.872**	0.830	**0.832**	**0.861**
$\mathrm{CNN_{mld}}$	0.929	0.947	0.929	0.947	0.797	0.734	0.694	0.726	**0.808**	0.764	**0.783**	0.759	0.867	0.846	**0.831**	0.844
$\mathrm{BLSTMCNN_m}$	0.917	0.931	0.917	0.931	0.763	0.696	0.637	0.684	0.760	0.755	0.734	0.755	0.863	0.846	0.826	0.843
$\mathrm{BLSTMCNN_{ml}}$	0.935	0.951	0.935	0.951	0.794	0.699	0.693	0.691	0.761	0.747	0.722	0.747	0.816	0.835	0.744	0.840
$\mathrm{BLSTMCNN_{mld}}$	0.927	0.938	0.927	0.937	0.799	0.707	0.694	0.692	0.770	0.747	0.740	0.747	0.859	0.839	0.814	0.837
$\mathrm{CBLSTM_m}$	0.940	**0.960**	0.940	**0.960**	0.799	0.721	0.696	0.723	0.781	0.774	0.754	0.774	0.866	0.840	0.830	0.838
$\mathrm{CBLSTM_{ml}}$	0.938	0.959	0.938	0.959	0.807	0.745	0.712	0.737	0.788	**0.787**	0.747	**0.785**	0.856	0.833	0.815	0.847
$\mathrm{CBLSTM_{mld}}$	0.936	0.947	0.937	0.947	0.809	0.745	0.714	0.741	0.791	0.779	0.761	0.778	0.863	**0.852**	0.825	0.849

Table 2: In-domain and multi-domain performance for all datasets.

4.5.3. Multi-domain results

The third series of tests was performed for the joint dataset WCL&W00&WFM (denoted by MULTI in Table 2 and Figure 3) with random 66%-33% split for training and test data, respectively (keeping the proportions of mixed data). Results are given in the last four columns of Table 2 and four last series in Figure 3.

As it can be seen, $\mathrm{CNN_{ml}}$ and $\mathrm{CNN_{mld}}$ outperform other models in the joint dataset, supporting the same conclusions as made for in-domain and cross-domain experiments.

4.6. Discussion

As can be observed from the experimental results, the proposed models outperform all baselines in all three scenarios (in-domain, cross-domain, and multi-domain). Also, comparing with results reported in (Anke and Schockaert, 2018), we can conclude that our models outperform SOA methods.

We also can confirm that the use of syntactic information as a part of input configurations (ml and mld) improves the results in most of scenarios. Moreover, during experiments we observe that when dependency encoding is used, keeping separate information about sentence words (as word vectors) has no significant impact on classification results. This allows us to decrease the model size and achieve better time complexity without harming performance significantly.

We can conclude that mathematical definitions require special treatment.

Finally, we see that generally both CNN and its combination with RNN is more beneficial than selecting a plain RNN model. This is probably due to the ability of CNN to perform abstract feature engineering before computing the classification model.

Unfortunately, no recommendation can be made about which word embedding to use for the DE task. Despite our expectations about better outcome with FastText vectors (due to its larger word coverage in three datasets), FastText not always provided better results.

We tried to understand which sentences represent difficult cases for our models. During annotation process, we found out that multiple sentences (two of them can be seen below) got different labels from different annotators. Finally, the label for such sentences was decided by major voting, but all annotators agreed that the decision was not unambiguous. We believe that most of false positive and false negative cases were created by such sentences and intend to explore

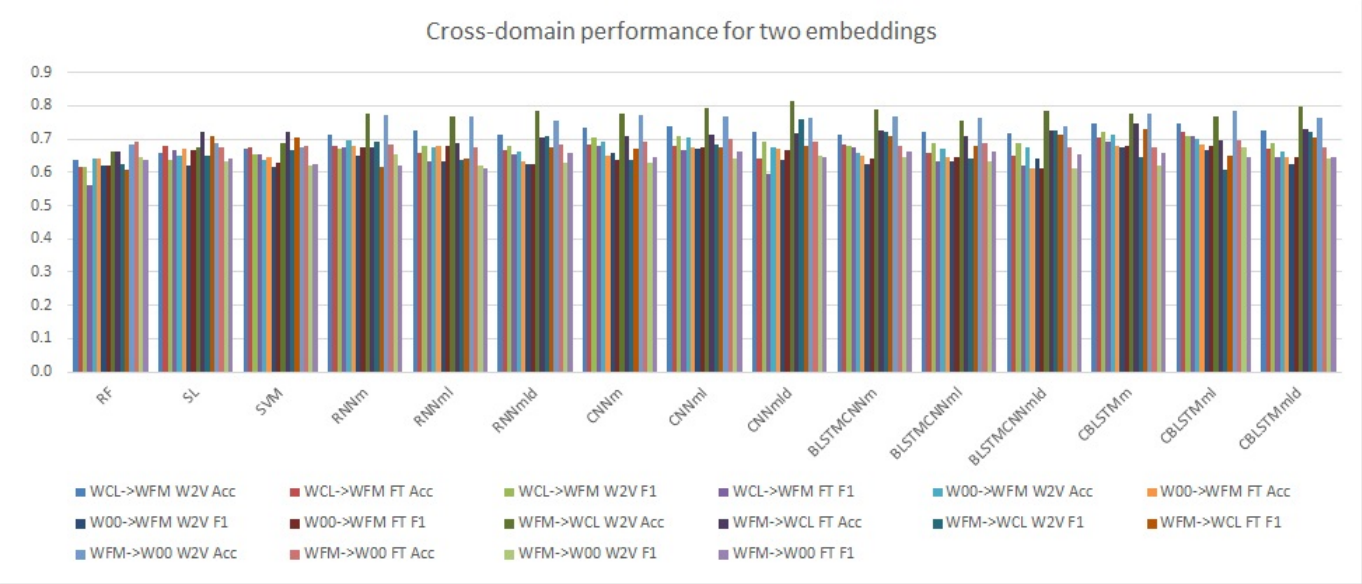

Figure 4: Cross-domain performance.

Dataset	WCL->**WFM**				W00->**WFM**				**WFM**->WCL				**WFM**->W00			
Method	W2V acc	FT acc	W2V F1	FT F1	W2V acc	FT acc	W2V F1	FT F1	W2V acc	FT acc	W2V F1	FT F1	W2V acc	FT acc	W2V F1	FT F1
RF	0.628	0.615	0.415	0.562	0.630	0.641	0.446	0.62	0.628	0.662	0.525	0.607	0.728	0.69	0.597	0.639
SL	0.573	0.679	0.415	0.668	0.572	0.669	0.388	0.668	0.654	0.721	0.565	0.71	0.713	0.676	0.578	0.643
SVM	0.625	0.675	0.412	0.654	0.628	0.646	0.404	0.627	0.648	0.722	0.558	0.706	0.726	0.679	0.591	0.623
RNN_{m}	0.713	0.680	0.672	0.676	0.697	0.678	0.651	0.675	0.778	0.676	0.692	0.616	0.773	0.685	0.654	0.621
RNN_{ml}	0.724	0.657	0.679	0.633	0.675	0.680	0.634	**0.680**	0.768	0.688	0.637	0.641	0.769	0.675	0.622	0.613
RNN_{mld}	0.712	0.668	0.681	0.653	0.663	0.632	0.625	0.625	0.785	0.703	0.710	0.674	0.755	0.685	0.629	0.657
CNN_{m}	0.734	0.685	0.705	0.679	0.694	0.651	0.660	0.638	0.778	0.709	0.635	0.670	0.773	0.691	0.627	0.644
CNN_{ml}	0.737	0.681	0.708	0.666	0.704	0.676	0.673	0.676	0.792	0.711	0.684	0.674	0.768	**0.701**	0.640	**0.663**
CNN_{mld}	0.723	0.643	0.694	0.597	0.676	0.669	0.636	0.668	**0.815**	0.716	**0.761**	0.681	0.765	0.692	0.650	0.648
$\text{BLSTMCNN}_{\text{m}}$	0.715	0.682	0.680	0.674	0.659	0.651	0.623	0.643	0.789	0.725	0.723	0.710	0.766	0.681	0.645	0.661
$\text{BLSTMCNN}_{\text{ml}}$	0.722	0.660	0.686	0.635	0.672	0.646	0.635	0.647	0.753	0.707	0.640	0.681	0.765	0.687	0.631	0.660
$\text{BLSTMCNN}_{\text{mld}}$	0.718	0.649	0.686	0.619	0.675	0.613	0.639	0.611	0.782	0.727	0.726	0.714	0.740	0.673	0.613	0.652
CBLSTM_{m}	**0.748**	0.706	**0.721**	0.693	**0.712**	0.680	**0.676**	**0.680**	0.778	**0.745**	0.645	**0.728**	0.777	0.674	0.621	0.657
$\text{CBLSTM}_{\text{ml}}$	**0.748**	**0.720**	0.708	**0.710**	0.698	**0.685**	0.666	0.678	0.768	0.698	0.608	0.651	**0.783**	0.695	**0.676**	0.647
$\text{CBLSTM}_{\text{mld}}$	0.725	0.671	0.689	0.646	0.660	0.646	0.624	0.646	0.799	0.729	0.721	0.703	0.766	0.675	0.643	0.645

Table 3: Cross-domain scores for all datasets.

more about that in our future error analysis.

Sentence 1 (annotated as definition):
An extremum may be local (a.k.a. a relative extremum; an extremum in a given region which is not the overall maximum or minimum) or global.

Sentence 2 (annotated as non-definition):
Exponential growth is common in physical processes such as population growth in the absence of predators or resource restrictions (where a slightly more general form is known as the law of growth).

5. Conclusions

In this paper we introduce a framework for DE from mathematical texts, using deep neural networks. We propose a novel representation for text sentences based solely on the dependency information, and its combination with deep neural networks, to handle the task. We also introduce a new annotated dataset of mathematical definitions, called WFM.

Our experiments demonstrate the superiority of CNN and its combination with RNN, applied on the syntactically-enriched input representation.

Also, we concluded that mathematical definitions require special treatment, and that using cross-domain learning for detection of mathematical definitions is inefficient.

In our future work, we plan to extend syntactic structure-based representation from single sentences to the surrounding text. We also plan to expand the classic DE task to the task of finding both definitions and sentences that reference these definitions in the same text.

Acknowledgment

The authors are grateful to Evgeny Naftaliev for help in the preparation and the annotation of the WFM dataset.

Bibliographical References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Anke, L. E. and Schockaert, S. (2018). Syntactically aware neural architectures for definition extraction. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), volume 2, pages 378–385.

Anke, L. E., Saggion, H., and Ronzano, F. (2015). Weakly supervised definition extraction. In Proceedings of the International Conference Recent Advances in Natural Language Processing, pages 176–185.

Bird, S., Dale, R., Dorr, B. J., Gibson, B. R., Joseph, M. T., Kan, M., Lee, D., Powley, B., Radev, D. R., and Tan, Y. F. (2008). The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco.

Boella, G. and Di Caro, L. (2013). Extracting definitions and hypernym relations relying on syntactic dependencies and support vector machines. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), volume 2, pages 532–537.

Borg, C., Rosner, M., and Pace, G. (2009). Evolutionary algorithms for definition extraction. In Proceedings of the 1st Workshop on Definition Extraction, pages 26–32. Association for Computational Linguistics.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Edwards, B. S. and Ward, M. B. (2004). Surprises from mathematics education research: Student (mis) use of mathematical definitions. *The American Mathematical Monthly*, 111(5):411–424.

Edwards, B. and Ward, M. B. (2008). Undergraduate mathematics courses. *Making the connection: Research and teaching in undergraduate mathematics education*, 73:223.

Edwards, B. (1998). Undergraduate mathematics majors' understanding and use of formal definitions in real analysis. *PhD thesis*.

Espinosa-Anke, L. and Saggion, H. (2014). Applying dependency relations to definition extraction. In International Conference on Applications of Natural Language to Data Bases/Information Systems, pages 63–74. Springer.

Fahmi, I. and Bouma, G. (2006). Learning to identify definitions using syntactic features. In Proceedings of the Workshop on Learning Structured Information in Natural Language Applications.

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

Honnibal, M. and Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1373–1378, Lisbon, Portugal, September. Association for Computational Linguistics.

Jin, Y., Kan, M.-Y., Ng, J.-P., and He, X. (2013). Mining scientific terms and their definitions: A study of the acl anthology. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 780–790.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Klavans, J. L. and Muresan, S. (2001). Evaluation of the definder system for fully automatic glossary construction. In Proceedings of the AMIA Symposium, page 324. American Medical Informatics Association.

Landau, S. I. (2001). Dictionaries: The art and craft of lexicography. Cambridge University Press, 2nd edition.

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

Li, S., Xu, B., and Chung, T. L. (2016). Definition extraction with lstm recurrent neural networks. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Springer, pp. 177–189.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pages 55–60.

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

Navigli, R. and Velardi, P. (2010). Learning word-class lattices for definition and hypernym extraction. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1318–1327. Association for Computational Linguistics.

Navigli, R., Velardi, P., Ruiz-Martínez, J. M., et al. (2010). An annotated dataset for extracting definitions and hypernyms from the web. In LREC.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Reiplinger, M., Schäfer, U., and Wolska, M. (2012). Extracting glossary sentences from scholarly articles: A comparative evaluation of pattern bootstrapping and deep analysis. In Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries, pages 55–65. Association for Computational Linguistics.

Robinson, R. (1962). Definitions. Oxford University Press, 1st edition.

Saggion, H. and Gaizauskas, R. J. (2004). Mining on-line

sources for definition knowledge. In FLAIRS Conference, pages 61–66.

Schuyler, P. L., Hole, W. T., Tuttle, M. S., and Sherertz, D. D. (1993). The umls metathesaurus: representing different views of biomedical concepts. *Bulletin of the Medical Library Association*, 81(2):217.

Storrer, A. and Wellinghoff, S. (2006). Automated detection and annotation of term definitions in german text corpora. In Proceedings of LREC, volume 2006.

Van Dormolen, J. and Zaslavsky, O. (2003). The many facets of a definition: The case of periodicity. *The Journal of Mathematical Behavior*, 22(1):91–106.

Weisstein, E. et al. (2007). Wolfram mathworld. `http://mathworld.wolfram.com/`.

Westerhout, E., Monachesi, P., and Westerhout, E. (2007). Combining pattern-based and machine learning methods to detect definitions for elearning purposes. In Proceedings of RANLP 2007 Workshop "Natural Language Processing and Knowledge Representation for eLearning Environments.

Westerhout, E. (2009). Definition extraction using linguistic and structural features. In Proceedings of the 1st Workshop on Definition Extraction, pages 61–67. Association for Computational Linguistics.

Language Resource References

Google. (2013). word2vec. `code.google.com/archive/p/word2vec/`.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Fasttext word vectors. `https://fasttext.cc/docs/en/crawl-vectors.html`.

Jin, Y., Kan, M.-Y., Ng, J.-P., and He, X. (2013). W00 definitions dataset. `https://bitbucket.org/luisespinosa/neural_de/src/afedc29cea14241fdc2fa3094b08d0d1b4c71cb5/data/W00_dataset/?at=master`.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D.). Python interface to corenlp using a bidirectional server-client interface. `https://github.com/stanfordnlp/python-stanford-corenlp`.

Navigli, R., Velardi, P., Ruiz-Martínez, J. M., et al. (2010). WCL definitions dataset. `http://lcl.uniroma1.it/wcl/`.

Vanetik, N., Litvak, M., Shevchuk, S., and Reznik, L. (2019). WFM dataset of mathematical definitions. `https://github.com/uplink007/FinalProject/tree/master/data/wolfram`.