mathematics

MDPI

Article

Definition Extraction from Generic and Mathematical Domains with Deep Ensemble Learning

Natalia Vanetik *,† **and Marina Litvak** *,†

Software Engineering Department, Shamoon College of Engineering, Bialik 56, Beer Sheva 8434231, Israel
* Correspondence: natalyav@sce.ac.il (N.V.); marinal@ac.sce.ac.il (M.L.)
† These authors contributed equally to this work.

Abstract: Definitions are extremely important for efficient learning of new materials. In particular, mathematical definitions are necessary for understanding mathematics-related areas. Automated extraction of definitions could be very useful for automated indexing educational materials, building taxonomies of relevant concepts, and more. 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. In this paper, we focus on automatic detection of one-sentence definitions in mathematical and general texts. We experiment with different classification models arranged in an ensemble and applied to a sentence representation containing syntactic and semantic information, to classify sentences. Our ensemble model is applied to the data adjusted with oversampling. Our experiments demonstrate the superiority of our approach over state-of-the-art methods in both general and mathematical domains.

Keywords: definition extraction; deep learning; ensemble; mathematical domain

Citation: Vanetik, N.; Litvak, M. Definition Extraction from Generic and Mathematical Domains with Deep Ensemble Learning. *Mathematics* **2021**, *9*, 2502. https://doi.org/10.3390/math9192502

Academic Editors: Cornelia Caragea and Florentina Hristea

Received: 1 September 2021
Accepted: 1 October 2021
Published: 6 October 2021

1. Introduction

Definitions play a very important role in scientific and educational literature because they define the major concepts that are operated inside the text. Despite mathematical and generic definitions being pretty similar in their linguistic style (see the example of two definitions below: the first, defining ASCII, is general, while the second defines mathematical object), supervised identification of mathematical definitions benefits from a training on a mathematical domain, as we previously showed in [1].

Definition 1. *American Standard Code for Information Interchange, also called ASCII, is a character encoding based on English alphabet.*

Definition 2. *The magnitude of a number, also called its absolute value, is its distance from zero.*

Naturally, we expect to find mathematical definitions in mathematical articles, which frequently use formulas and notations in both definitions and surrounding text. The number of words in mathematical text is smaller than in standard text due to the formulas that are used to express the former. The mere presence of formulas is not a good indicator of a definition sentence because the surrounding sentences may also use notations and formulas. As an example of such text, Definition 3, below, contains a definition from Wolfram MathWorld. Only the first sentence in this text is considered a definition sentence, even though other sentences also contain mathematical notations.

Definition 3. *A finite field is a field with a finite field order (i.e., number of elements), also called a Galois field. The order of a finite field is always a prime or a power of a prime. For each prime power, there exists exactly one (with the usual caveat that "exactly one" means "exactly one up to an isomorphism") finite field $GF(p^n)$, often written as $\mathbb{F}_{p^n}$ in current usage.*

Definition extraction (DE) is a challenging and popular task today, as shown by a recent research call at SemEval-2020 shows (https://competitions.codalab.org/competitions/20900, accessed on 1 September 2021).

Multiple 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 applied for this task, employing feature engineering for sentence representation. However, all recently published works study generic definitions, without evaluation of their methods on mathematical texts.

In this paper, we describe a supervised learning method for automatic DE from both generic and mathematical texts. Our method applies ensemble learning to adjusted-by-oversampling data, where 12 deep neural network-based models are trained on a dataset with labeled definitions and then applied on test sentences. The final label of a sentence is decided by the ensemble voting.

Our method is evaluated on four different corpora; three for generic DE and one is an annotated corpus of mathematical definitions.

The main contributions of this paper are (1) the introduction of a new corpus of mathematical texts with annotated sentences and (2) an evaluation of an ensemble learning model for the DE task, (3) an evaluation of the introduced ensemble learning model on a general and mathematical domains, including (4) cross-domain experiments. We performed extensive experiments with different ensemble models on four datasets, including the introduced one. Our experiments demonstrate the superiority of our model for the three out of four datasets, belonging to two different domains. The paper is organized as follows. Section 2 contains a survey of up-to-date related work. Section 3 describes our approach. Section 4 provides the evaluation results and their analysis. Finally, Section 5 contains our conclusions.

2. Related Work

Definition extraction has been a popular topic in NLP research for more than a decade [2], and it remains a challenging and popular task today as a recent research call at SemEval-2020 show. 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. Klavans and Muresan [3] presented the DEFINDER, a rule-based system that mines consumer-oriented full text articles to extract definitions and the terms they define; the system is evaluated on definitions from on-line dictionaries such as the UMLS Metathesaurus [4]. Xu et al. [2] used various linguistic tools to extract kernel facts for the definitional question-answering task in Text REtrieval Conference (TREC) 2003. Malaise et al. [5] used semantic relations to mine defining expressions in domain-specific corpora, thus detecting semantic relations between the main terms in definitions. This work is evaluated on corpora from fields of anthropology and dietetics. Saggion and Gaizauskas [6], Saggion [7] employed analysis of on-line sources to find lists of relevant secondary terms that frequently occur together with the definiendum in definition-bearing passages. Storrer and Wellinghoff [8] proposed a system that automatically detects and annotates definitions for technical terms in German text corpora. Their approach focuses on verbs that typically appear in definitions by specifying search patterns based on the valency frames of definitor verbs. Borg et al. [9] extracted definitions from nontechnical texts using genetic programming to learn the typical linguistic forms of definitions and then using a genetic algorithm to learn the relative importance of these forms. Most of these methods suffer from both low recall and precision (below 70%), because definition sentences occur in highly variable and noisy syntactic structures.

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 [10] presented an approach to learning concept definitions from fully parsed text with a maximum entropy classifier incorporating various syntactic features; they tested this approach on a subcorpus of the Dutch version of Wikipedia. In [11], a pattern-based glossary candidate detector, which is capable of extracting definitions in eight languages, was presented. Westerhout [12] described a combined approach that first filters corpus with a definition-oriented grammar, and then applies machine learning to improve the results obtained with the grammar. The proposed algorithm was evaluated on a collection of Dutch texts about computing and e-learning. Navigli and Velardi [13] used Word-Class Lattices (WCLs), a generalization of word lattices, to model textual definitions. The authors introduced a new dataset called WCL that was used for the experiments. They achieved a 75.23% F1 score on this dataset. Reiplinger et al. [14] compared lexico-syntactic pattern bootstrapping and deep analysis. The manual rating experiment suggested that the concept of definition quality in a specialized domain is largely subjective, with a 0.65 agreement score between raters. The DefMiner system, proposed in [15], used Conditional Random Fields (CRF) to predict the function of a word and to determine whether this word is a part of a definition. The system was evaluated on a W00 dataset [15], which is a manually annotated subset of ACL Anthology Reference Corpus (ACL ARC) ontology. Boella and Di Caro [16] proposed a technique that only uses syntactic dependencies between terms extracted with a syntactic parser and then transforms syntactic contexts to abstract representations to use a Support Vector Machine (SVM). Anke et al. [17] proposed a weakly supervised bootstrapping approach for identifying textual definitions with higher linguistic variability. Anke and Saggion [18] presented a supervised approach to DE in which only syntactic features derived from dependency relations are used.

Algorithms in the third category use Deep Learning (DL) techniques for DE, often incorporating syntactic features into the network structure. Li et al. [19] used Long Short-Term Memory (LSTM) and word vectors to identify definitions and then tested this approach on the English and Chinese texts. Their method achieved a 91.2% F-measure on the WCL dataset. Anke and Schockaert [20] combined Convolutional Neural Network (CNN) and LSTM, based on syntactic features and word vector representation of sentences. Their experiments showed the best F1 score (94.2%) on the WCL dataset for CNN and the best F1 score (57.4%) on the W00 dataset for the CNN and Bidirectional LSTM (BLSTM) combination, both with syntactically enriched sentence representation. Word embedding, when used as the input representation, have been shown to boost the performance in many NLP tasks, due to its ability to encode semantics. We believe that a choice to use word vectors as input representation in many DE works was motivated by its success in NLP-related classification tasks.

We use the approach of [20] as a starting point and as a baseline for our method. We further extend this work by (1) additional syntactic knowledge in a sentence representation model, (2) testing additional network architectures, (3) combining 12 configurations (that were the result of different input representations and architectures) in a joint ensemble model, and (4) evaluation of the proposed methodology on a new dataset of mathematical texts. As previously shown in our and others' works [1,20], dependency parsing can add valuable features to the sentence representation in the DE task, including mathematical DE. The same works showed that a standard convolutional layer can be sufficiently applied to automatically extract the most significant features from the extended representation model, which improves accuracy for the classification task. Word embedding matrices enhanced with dependency information naturally call for CNN due to their size and CNN's ability to decrease dimensionality swiftly. On the other hand, sentences are sequences for which LSTM is naturally suitable. In [1], we explored how the order of the CNN and LSTM layers and the input representation affect the results. To do that, we evaluated and compared between 12 configurations (see Section 3.3). As result, we obtained a conclusion that CNN and its combination with LSTM, applied on a syntactically enriched input representation,

outperform other configurations. However, as we show in our experimental evaluation (see Section 4), all the models individually are inferior to the ensemble approach.

Following recent research demonstrating the superiority of pretrained language models on many NLP tasks, including DE [21,22], we apply fine-tuned BERT [23,24] on our data and compare its results with the results of the proposed method.

3. Method

This section describes out method, including representation of input sentences, individual composition models, and their combination through ensemble learning.

3.1. Sentence Representation

First, we generate a sentence matrix from word vectors of its words as follows. Every word w is represented by its $k = 300$-dimensional word vector $\vec{w}$ [25], and all sentences are assumed as having the same length n, using zero padding where necessary; as a result, we obtain a sentence matrix denoted by $S_{n\times k}$.

Then, we generate the following three syntactically enriched representations:

1. **(m)**—an extension of matrix $S_{n\times k}$ with the dependency information, where a dependency is represented by the average of word vectors of the two words participating in it as:
$$m = S_{n\times k} \circ [\vec{r_{ij}}^{avg}]_{ij}$$
2. **(ml)**—an extension of (m) with one-hot encoding of dependency labels between pairs of words [20]; formally, it includes the average of word vectors of dependency words, and dependency label representations as follows:
$$ml = S_{n\times k} \circ [\vec{r_{ij}}^{avg} \circ dep_{ij}]_{ij}$$
3. **(mld)**—it is composed of the full dependency information, including concatenation of word vectors for dependency words, dependency label, and dependency depth. In contrast to the first two representations, (mld) does not contain the matrix $S_{n\times k}$ itself, but only the dependency information as follows:
$$mld = [\vec{w_i} \circ \vec{w_j} \circ dep_{ij} \circ depth_{ij}]_{ij}$$

The dependency representations in the input configurations described above, are depicted in Figure 1. All input vectors are zero-pad to compensate for vector size differences. In all representations we use fastText vectors pretrained on English webcrawl and Wikipedia [26], based on our observation in [1] about superiority of fastText vectors over other pretrained word vectors.

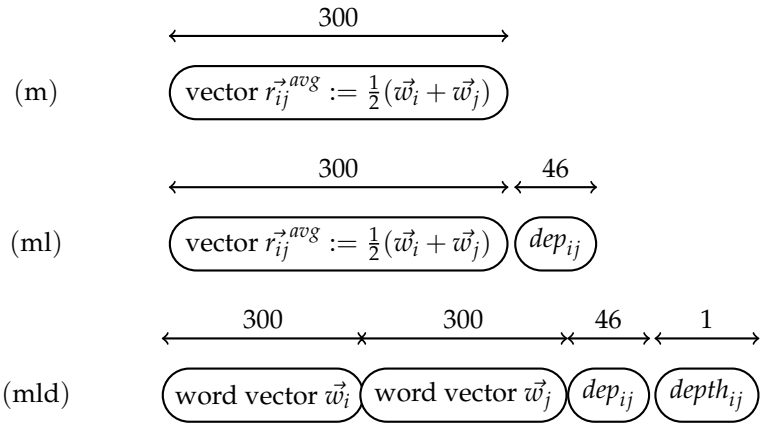

Figure 1. Dependency representation for input configurations (m), (ml) and (mld).

3.2. Composition NN Models

We use the approach of Anke and Schockaert [20] as a starting point and as a baseline for our method. We further extend this work by additional syntactic knowledge in a sentence representation model, and by additional changes in network architectures, based on [1]. We experiment with two layers: convolutional layer which can help to automatically extract the most significant features before performing the classification task (given big matrices representing sentences with word vectors and syntax features) and Bidirectional LSTM layer which is very suitable for sentences as sequences. We use four neural models:

1. Pure 2-dimensional CNN model,
2. Pure Bidirectional LSTM model,
3. Mixed model with a CNN layer followed by a Bidirectional LSTM layer, denoted by CNN_LSTM (see Figure 2),
4. Mixed model with a Bidirectional LSTM layer followed by a CNN layer, denoted by LSTM_CNN (see Figure 3).

Every one of those models is used in conjunction with the three sentence representations described in Section 3.1, giving us total of 12 models.

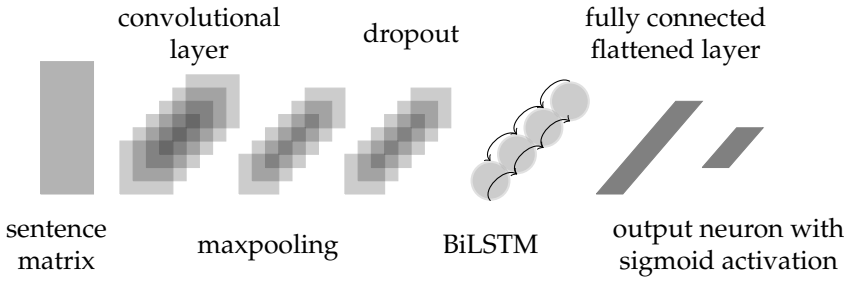

Figure 2. CNN_LSTM network architecture.

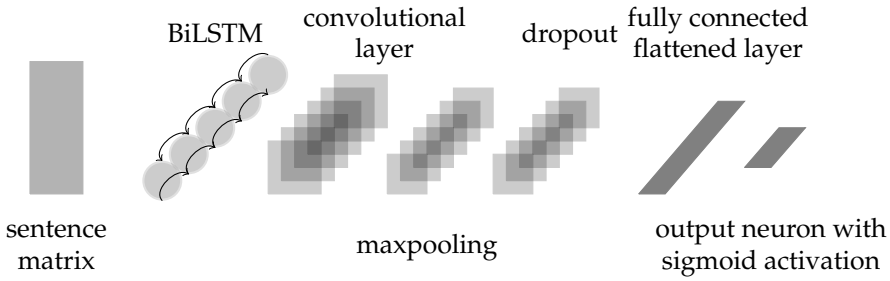

Figure 3. LSTM_CNN network architecture.

The intuition behind using mixed models is supported by known benefits of their composition layers—CNN can automatically extract features, while LSTM is a classification model that is context-aware. The experiment, performed in [1], was aimed to examine which order of layers is beneficial for the DE task—first to extract features from the original input and then feed them to the context-aware classifier, or first to calculate hidden states with context-aware LSTM gates and then feed them into CNN classifier for feature extraction before the classification layer. The results demonstrated the superiority of models with a CNN layer, 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, allowing the network to be more accurate with fewer parameters. Due to a high-dimensional input in our task (also in context-aware representation, produced by LSTM gates), this characteristic of CNN appears to be very helpful.

3.3. Ensemble Pipeline

Given two basic models, two different combinations of CNN and LSTM layers, and three variations of the sentence representation, we finally obtain 12 different models which are trained and then applied separately on the test sentences. We have applied oversampling with a 'minority' setting to a dataset (It is worth noting that a general practice of changing class weights in all the models did not result in accuracy improvement in our experiments.). To produce a final label for each test sentence, ensemble voting was applied. Figure 4 depicts the pipeline of our approach. We tried different supervised models for ensemble voting, which we describe in the Experiments section. The dataset (denoted as data X in Figure 4) was split in the following manner: 70% and 5% of the dataset was used for training and validation of the individual models, respectively. Then, the trained individual models were applied on 25% of the X data (denoted as Y in Figure 4), while the entire Y data were further split as follows: 5% of the X data (or 20% of Y) were used as a training set and 20% of X (equals to 80% of Y) as a test set for the ensemble model.

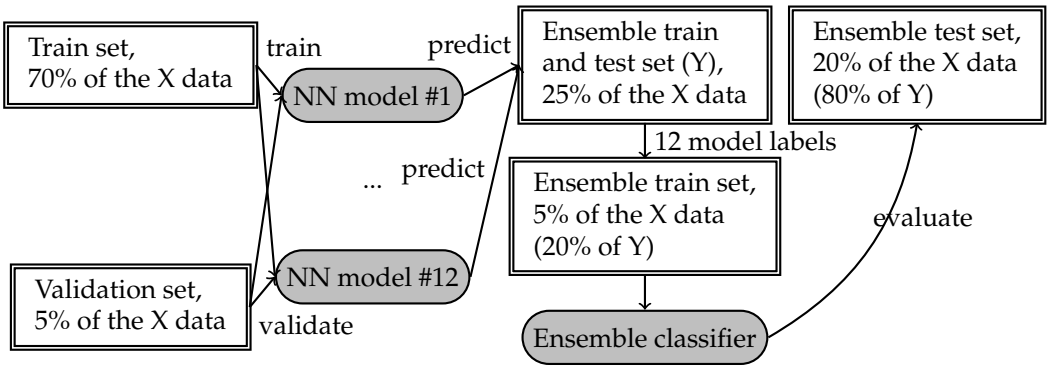

Figure 4. Pipeline of our approach for ensemble learning.

To see the advantage of our ensemble approach, we also evaluated every one of the 12 neural models individually on the data after oversampling. The pipeline of individual model evaluation is depicted in Figure 5.

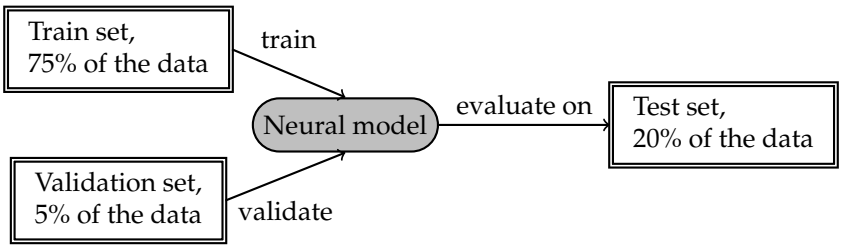

Figure 5. Evaluation pipeline for individual NN models.

4. Experiments

Our experiments aim at testing our hypothesis about superiority of the ensemble of all 12 composition models over single models and SOTA baselines.

Based on the feature analysis described in Section 4.8, we decided to employ all 12 configurations as composition models and, based on the observations from our previous work [1], we use pretrained fastText (further denoted by FT) word embedding in all of them.

Tests were performed on a cloud server with 32 GB of RAM, 150 GB of PAGE memory, an Intel Core I7-7500U 2.70 GHz CPU, and two NVIDIA GK210GL GPUs.

4.1. Tools

The models were implemented with help of the following tools: (1) Stanford CoreNLP wrapper [27] for Python (tokenization, sentence boundary detection, and dependency parsing), (2) Keras [28] with Tensorflow [29] as a back-end (NN models), (3) fastText vectors

pretrained on English webcrawl and Wikipedia [26], (4) Scikit-Learn [30] (evaluation with F1, recall, and precision metrics), (5) fine-tunable BERT python package [24] available at https://github.com/strongio/keras-elmo (accessed on 1 September 2021), and (6) WEKA software [31]. All neural models were trained with batch size 32 and 10 epochs.

4.2. Datasets

In our work we use the following four datasets–DEFT, W00, WCL, and WFMALL–that are described below. The dataset domain, number of sentences for each class, majority vote values, total number of words, and number of words with non-zero fastText (denoted by FT) word vectors are given in Table 1.

Table 1. Dataset statistics.

Dataset	Domain	Definitions	Non-Definitions	Majority	Words	Covered by FT
WCL	General	1871	2847	0.603	21,297	16,645
DEFT	General	562	1156	0.673	7644	7350
W00	General	731	1454	0.665	8261	7003
WFMALL	Math	1934	4206	0.685	13,138	8238

4.2.1. The WCL Dataset

The World-Class Lattices (WCL) dataset [32] was introduced in [33]. It is constructed from manually annotated Wikipedia articles in English. We used the WCL v1.2 version that contains 4719 annotated sentences, 1871 of which are proper definitions and 2847 are *distractor* sentences that have similar structures with proper definitions, but are not actually definitions. This dataset focuses on generic definitions in various areas. 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.*

WCL contains the following annotations: (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 dataset description, existence of the first three fields indicates that a sentence is a definition.

4.2.2. The DEFT Dataset

The DEFT corpus, proposed in [34], consists of annotated content from two different data sources: various 2017 SEC contract filings from the publicly available US Securities and Exchange Commission EDGAR (SEC) database, and sentences from the https://cnx.org/ open source textbooks (accessed on 1 September 2021). The partial corpus is available for download from GitHub at https://github.com/adobe-research/deft_corpus (accessed on 1 September 2021). We have used this part of the DEFT corpus in our experiments; it contains 562 definition sentences and 1156 non-definition sentences. A sample definition sentence from this dataset is

> *A hallucination is a perceptual experience that occurs in the absence of external stimulation.*

and a sample non-definition sentence is

In monocots, petals usually number three or multiples of three; in dicots, the number of petals is four or five, or multiples of four and five.

4.2.3. The W00 Dataset

The W00 dataset [35], introduced in [15], was compiled from ACL ARC ontology [36] and contains 2185 manually annotated sentences, with 731 definitions and 1454 non-definitions; 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 not to be a definition if all its tokens are marked as *O*. Sentence that contains tokens marked as *T* or *D* is considered to be a definition.

4.2.4. The WFMALL Dataset

The WFMALL dataset is an extension of the WFM dataset [37], introduced by us in [1]. It was created by us after collecting and processing all 2352 articles from Wolfram Mathworld [38]. The final dataset contains 6140 sentences, of which 1934 are definitions and 4206 are non-definitions. Sentences were extracted automatically and then manually separated into two categories: definitions and statements (non-definitions). All annotators (five in total) have at least BSc degree and learned academic mathematical courses (research group members, including three research students). The data were semi-automatically segmented to sentences with Stanford CoreNLP package and then manually assessed. All malformed sentences (as result of wrong segmentation) were fixed, 116 very short sentences (with less than 3 words) were removed. All sentences related to Wolfram Language (https://en.wikipedia.org/wiki/Wolfram_Language, accessed on 1 September 2021) were removed because they relate to a programming language and describe how mathematical objects are expressed in this language, and not how they are defined. Sentences with formulas only, without text, were also removed. The final dataset was split to nine portions, saved as Unicode text files. Three annotators worked on each portion. First, two annotators labeled sentences independently. Then, all sentences that were given different labels were finally annotated by the third annotator (controller) (We decided that a label with majority vote will be selected. Therefore, the third annotator (controller) labeled only the sentences with contradict labels.). The final label was set by majority vote. The Kappa agreement between annotators was 0.651, which is considered substantial agreement.

This dataset is freely available for download from GitHub (https://drive.google.com/drive/folders/1052akYuxgc2kbHH8tkMw4ikBFafIW0tK?usp=sharing, accessed on 1 September 2021). A sample definition 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. Evaluation Setup

Datasets were oversampled with the 'minority' setting, and split to the NN training set (70%), the NN validation set (5%), and the NN test set (25%). Then the labels of all 12 models on the NN test set were used to form the ensemble dataset, which was further split into the ensemble training set (5% of the original data size) and the ensemble test set

(20% of the original data size). In this setting, we have a test set size identical to those of baselines that used 75% training, 5% validation, and 20% test set split.

For the consistency of the experiment, it was important to us to keep the same training and validation sets for individual models, whether as stand-alone or as composition models in an ensemble. Additionally, all evaluated and compared models, including ensemble, were evaluated on the same test set. Because ensemble models were trained using traditional machine-learning algorithms, no validation set was needed for them.

4.4. Text Preprocessing

Regarding all four 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: DEFT, WCL, and W00 datasets came pre-split, and sentence splitting for the new WFMALL dataset was performed semi-automatically by our team (using Stanford CoreNLP SBD, followed by manual correction, due to many formulas in the text).
- Tokenization and dependency parsing were performed on all sentences with the help of the Stanford CoreNLP package [39].
- For the W00 datasets used in [20] for DE, we replaced parsing by SpaCy [40] with the Stanford CoreNLP parser.
- We applied fastText [41] vectors pretrained on English webcrawl and Wikipedia (available at [26]).

4.5. Baselines

We compared our results with five baselines:

- DefMiner [15], which uses Conditional Random Fields to detect definition words;
- BERT [23],[24], fine-tuned on the training subset of every dataset for the task of sentence classification;
- CNN_LSTM$_{ml}$, proposed in [20];
- CNN_LSTM$_{m}$, which is the top-ranked composition model on the adjusted WFMALL dataset.
- CNN_LSTM$_{mld}$, which is the top-ranked composition model on the adjusted W00 dataset.

We applied two supervised models for the ensemble voting: Linear Regression (LR) and Random Forest (RF). The results reported here are the average over 10 runs, with random reshuffling applied to the dataset each time (We did not apply a standard 10-fold cross validation due to a non-standard proportion between a training and a test dataset. Additionally, 10-fold cross validation was not applied on individual models, as it is not a standard evaluation technique for deep NNs.). We also applied the majority voting, denoted as Ensemble majority (Our code will be released once the paper is accepted).

4.6. Results for the Mathematical Domain

Table 2 contains the evaluation results (accuracy) for all systems on the WFMALL dataset, with bold font indicating the best scores. It can be seen that (1) oversampling improves performance of NN baselines and (2) all ensemble models outperform the baseline systems and all the individual NN models.

Table 2. Final results for the WFMALL dataset.

Baseline	**Oversample**	**Accuracy**
Fine-tuned BERT	no	0.760
Fine-tuned BERT	yes	0.750
CNN_LSTM$_m$	no	0.860
CNN_LSTM$_{ml}$	no	0.864
CNN_LSTM$_{mld}$	no	0.867
DefMiner	N\A	0.704
Model	**Oversample**	**Accuracy**
CNN$_m$	yes	0.909
CNN$_{ml}$	yes	0.913
CNN$_{mld}$	yes	0.884
CNN_LSTM$_m$	yes	**0.922**
CNN_LSTM$_{ml}$	yes	0.917
CNN_LSTM$_{mld}$	yes	**0.922**
LSTM$_m$	yes	0.884
LSTM$_{ml}$	yes	0.906
LSTM$_{mld}$	yes	0.916
LSTM_CNN$_m$	yes	0.901
LSTM_CNN$_{ml}$	yes	0.891
LSTM_CNN$_{mld}$	yes	0.909
Ensemble majority	yes	0.925
Ensemble LR	yes	0.937
Ensemble RF	yes	**0.943**

4.7. Results for the General Domain

To see that our approach may be used for general definitions as well, we have chosen the WCL dataset [15], the W00 dataset [15] and the DEFT dataset [34]. The DefMiner system (used here as one of the baselines) is the system that was first applied to W00, and to this day it produced the best results on it; several systems that were suggested in the literature [1,20,42], were unable to outperform the DefMiner system on W00.

Tables 3–5 contain the evaluation results (accuracy) for all systems on the W00, the DEFT, and the WCL datasets, respectively; the best results are marked in bold. The tables show that (1) oversampling significantly improves performance of NN baselines, resulting in superiority over DefMiner on W00, and (2) the ensemble models (at least one of them) significantly outperform all individual neural models and baselines, including DefMiner and fine-tuned BERT.

The WCL dataset is considered to be an 'easy' dataset with several proposed systems [20,34] reporting accuracy of over 0.98 on this dataset; however, none of the methods suggested in these works achieve similar results on more challenging datasets such as DEFT and W00.

Table 3. Final results for the W00 dataset.

Baseline	**Oversample**	**Accuracy**
Fine-tuned BERT	no	0.670
Fine-tuned BERT	yes	0.620
CNN_LSTM$_m$	no	0.709
CNN_LSTM$_{ml}$	no	0.716
CNN_LSTM$_{mld}$	no	0.705
DefMiner	N\A	0.819

Table 3. *Cont.*

Model	Oversample	Accuracy
CNN_m	yes	0.828
CNN_{ml}	yes	0.825
CNN_{mld}	yes	0.825
CNN_LSTM_m	yes	0.817
CNN_LSTM_{ml}	yes	0.799
CNN_LSTM_{mld}	yes	0.839
$LSTM_m$	yes	0.785
$LSTM_{ml}$	yes	0.783
$LSTM_{mld}$	yes	0.812
$LSTM\_CNN_m$	yes	0.772
$LSTM\_CNN_{ml}$	yes	0.791
$LSTM\_CNN_{mld}$	yes	0.783
Ensemble majority	yes	**0.879**
Ensemble LR	yes	0.850
Ensemble RF	yes	0.854

Table 4. Final results for the DEFT dataset.

Baseline	Oversample	Accuracy
Fine-tuned BERT	no	0.670
Fine-tuned BERT	yes	0.670
CNN_LSTM_m	no	0.719
CNN_LSTM_{ml}	no	0.732
CNN_LSTM_{mld}	no	0.717
DefMiner	N\A	0.710
Model	**Oversample**	**Accuracy**
CNN_m	yes	0.826
CNN_{ml}	yes	0.819
CNN_{mld}	yes	0.853
CNN_LSTM_m	yes	0.850
CNN_LSTM_{ml}	yes	0.839
CNN_LSTM_{mld}	yes	0.832
$LSTM_m$	yes	0.801
$LSTM_{ml}$	yes	0.786
$LSTM_{mld}$	yes	0.824
$LSTM\_CNN_m$	yes	0.789
$LSTM\_CNN_{ml}$	yes	0.824
$LSTM\_CNN_{mld}$	yes	0.822
Ensemble majority	yes	**0.867**
Ensemble LR	yes	0.850
Ensemble RF	yes	0.846

Table 5. Final results for the WCL dataset.

Baseline	Oversample	Accuracy
Fine-tuned BERT	no	0.94
Fine-tuned BERT	yes	0.95
CNN_LSTM_m	no	0.948
CNN_LSTM_{ml}	no	0.947
CNN_LSTM_{mld}	no	0.945
DefMiner	N\A	0.797

Table 5. *Cont.*

Model	Oversample	Accuracy
CNN_m	yes	0.955
CNN_{ml}	yes	0.965
CNN_{mld}	yes	0.958
CNN_LSTM_m	yes	0.964
CNN_LSTM_{ml}	yes	0.965
CNN_LSTM_{mld}	yes	0.956
$LSTM_m$	yes	0.950
$LSTM_{ml}$	yes	0.950
$LSTM_{mld}$	yes	0.959
$LSTM\_CNN_m$	yes	0.953
$LSTM\_CNN_{ml}$	yes	0.950
$LSTM\_CNN_{mld}$	yes	0.957
Ensemble majority	yes	**0.972**
Ensemble LR	yes	0.963
Ensemble RF	yes	**0.972**

4.8. Cross-Domain Results

We also conducted a cross-domain analysis using the best individual neural models and ensemble models for every dataset in the mathematical and the general domains (Table 6); the best scores are marked in bold.

The individual models were trained on data set X and tested on dataset Y, with X coming from one domain and Y from another, as depicted in Figure 6. In this case, we selected the individual neural model that was the most successful for the training dataset.

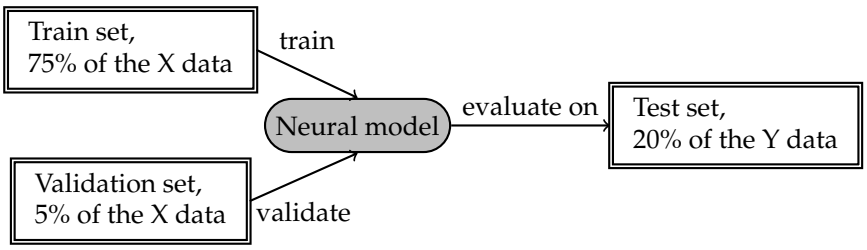

Figure 6. Pipeline of cross-domain evaluation of individual NN models for X (training) → Y (test) datasets.

For the ensemble evaluation, we trained all 12 models and the ensemble classifier using the same pipeline as in Section 3.3 on the training set (denoted as Dataset X in Figure 7). Then, these 12 models produce the labels for both 5% of the Dataset X (20% from its test set that is used for training ensemble weights) and 20% of the Dataset Y (its entire test set). The ensemble model is trained on 5% of the Dataset X and applied on 20% of the Dataset Y, using labels produced by 12 trained individual models.

To make this process fully compatible to an in-domain testing procedure (depicted in Figures 4 and 5), we used the same dataset splits both for evaluation of individual models (see Figure 6) and for evaluation of ensemble models (see Figure 7).

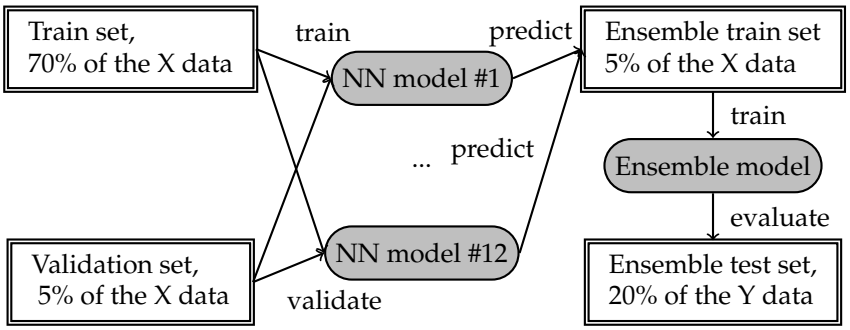

Figure 7. Pipeline of ensemble cross-domain evaluation for X (training) → Y (test) datasets.

Table 6. Cross-domain tests.

Training→Test	Model	Resampling	Accuracy
WCL→WFMALL	CNN_LSTM$_{ml}$	yes	**0.754**
	Ensemble LR	yes	0.633
	Ensemble RF	yes	0.661
	Ensemble majority	yes	0.656
W00→WFMALL	CNN_LSTM$_{mld}$	yes	0.575
	Ensemble LR	yes	0.638
	Ensemble RF	yes	0.651
	Ensemble majority	yes	**0.664**
DEFT→WFMALL	CNN$_{ml}$	yes	**0.761**
	Ensemble LR	yes	0.581
	Ensemble RF	yes	0.592
	Ensemble majority	yes	0.620
WFMALL→WCL	CNN_LSTM$_{m}$	yes	**0.832**
	Ensemble LR	yes	0.608
	Ensemble RF	yes	0.590
	Ensemble majority	yes	0.601
WFMALL→W00	CNN_LSTM$_{m}$	yes	0.700
	Ensemble LR	yes	0.666
	Ensemble RF	yes	0.691
	Ensemble majority	yes	**0.709**
WFMALL→DEFT	CNN_LSTM$_{m}$	yes	**0.738**
	Ensemble LR	yes	0.605
	Ensemble RF	yes	0.634
	Ensemble majority	yes	0.659

As we can see from Table 6, all cases where the training set comes from one domain and the test set is from another domain produce the significantly lower accuracy than that reported in Tables 2–5. This is a testament to the fact that general definition domain and mathematical domain are quite different.

4.9. Parameter Selection and Evaluation

Our work aimed to evaluate the effect of syntactic information for the task of definition extraction. The prior work [20] in this field has demonstrated that relying on the word information alone (such as word embedding) does not produce good results. Furthermore, classification accuracy depends heavily on the dataset—higher accuracy scores are achieved on the WCL dataset [32] by methods in [20] and [1] because both the word embeddings used for the task and the data originate in Wikipedia. On other datasets such as DEFT, W00, and our WFMALL accuracy scores of individual models (see Tables 2–5) are significantly lower.

To combat this situation, we experimented with different neural models and their parameters, such as the number of layers, type of layers, learning rate, and so on. However, the classification accuracy was only slightly affected by the change in these parameters—for

instance, increasing the number of training epochs above 15 did not improve the scores at all. The final parameters we used for our neural models appear in Table 7.

Table 7. NN parameters for the individual models.

Parameter	Value (s)
learning rate	0.01 (default)
number of layers and neurons	one convolutional layer for the CNN and CNN_LSTM, one bidirectional LSTM layer for LSTM and LSTM_CNN
activation	sigmoid for the last layer, ReLU for the other layers
loss	binary_crossentropy
regularization	0.01 for all regularization parameters
optimizer	Adam
dropout	0.5

Furthermore, we incorporated syntactic information into our models using input representations depicted in Figure 1. However, we observed that individual models rank differently on different datasets, and there is no clear advantage in using one specific syntactic representation because such a selection does not translate well across datasets. Therefore, we decided to use the ensemble model to compensate for variation in the data, and to use resampling to balance the data.

To analyze the importance and necessity of composition models in our ensemble, we performed feature selection by evaluating the information gain of labels produced by each model. The results are shown in Table 8 for the four datasets; the highest ranking attributes are shown in bold. As can be seen, with one exception per dataset, all models with the first or only CNN layer are ranked higher than other models. As such, we can conclude that models that use CNN as their first layer are more successful and have higher influence on the ensemble scores, but their exact influence also depends on a dataset. However, feature backward elimination showed that all 12 models are necessary and produce the best accuracy in ensemble. Eliminating individual models one by one from the ensemble produced less accurate ensemble models.

Table 8. Information gain rankings for individual models within the ensemble model.

Model	WFMALL	W00	DEFT	WCL
CNN_{m}	0.585	0.349	0.349	0.735
CNN_{ml}	0.594	0.341	0.341	0.781
CNN_{mld}	0.529	**0.423**	**0.423**	0.748
CNN_LSTM_{m}	**0.625**	0.393	0.393	0.780
CNN_LSTM_{ml}	0.606	0.368	0.368	**0.782**
CNN_LSTM_{mld}	0.609	0.362	0.362	0.741
$LSTM_{m}$	0.507	0.296	0.269	0.714
$LSTM_{ml}$	0.557	0.269	0.296	0.718
$LSTM_{mld}$	0.584	0.328	0.328	0.755
$LSTM\_CNN_{m}$	0.549	0.285	0.285	0.726
$LSTM\_CNN_{ml}$	0.538	0.346	0.346	0.715
$LSTM\_CNN_{mld}$	0.576	0.338	0.338	0.746

4.10. Error Analysis

We tried to understand which sentences represented difficult cases for our models. During annotation process, we found that multiple sentences were assigned different labels by different annotators. Finally, the label for such sentences was decided by majority voting, but all annotators agreed that the decision was not unambiguous. Based on our observation and manual analysis, we believe that most of the false positive and false negative cases were created by such sentences. We categorized these sentences to the following cases:

1. Sentences describing properties of a mathematical object. Example (annotated (gold standard label = "definition") 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.*

We did not instruct our annotators regarding labeling this sentence type and let them make decisions based on their knowledge and intuition. As result, this sentence type received different labels from different annotators.

2. Sentences providing alternative naming of a known (and previously defined) mathematical object. Example (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).*

   We received the same decisions and the same outcomes in our dataset with this sentence type as with type (1).
3. Formulations—sentences that define some mathematical object by a formula (in contrast to a verbal definition, that explains the object's meaning). Example (annotated as *non-definition*):

   > l *Formulas expressing trigonometric functions of an angle 2x in terms of functions of an angle x, sin(2x) = [FORMULA].*

   If both definition and formulation sentences for the same object were provided, our annotators usually assigned them different labels. However, rarely a mathematical object can be only defined by a formula. Additionally, sometimes it can be defined by both, but the verbal definition is not provided in an analyzed article. In such cases, annotators assigned the "definition" label to the formulation sentence.
4. Sentences that are parts of a multi-sentence definition. Example (annotated as *non-definition*):

   > *This polyhedron is known as the dual, or reciprocal.*

   We instructed our annotators not to assign "definition" label to sentences that do not contain comprehensive information about a defined object. However, some sentences were still annotated as "definition", especially when they appear in a sequence.
5. Descriptions—sentences that describe mathematical objects but do not define them unequivocally. Example (annotated as *non-definition*):

   > *A dragon curve is a recursive non-intersecting curve whose name derives from its resemblance to a certain mythical creature.*

   Although this sentence resembles a legitimate definition (grammatically), it was labeled as non-definition because its claim does not hold in both directions (not every recursive non-intersecting curve is a dragon curve). Because none of our annotators was expert in all mathematical domains, it was difficult for them to assign the correct label in all similar cases.

As result of subjective annotation (which occurs frequently in all IR-related areas), none of the ML models trained on our training data were very precise with the ambiguous cases such as those described above. Below are several examples of sentences misclassified as definitions (false positives (with gold standard label "non-definition" but classified as "definition")), from each type described in the list above:

1. Property description:

   > *Every pentagonal number is 1/3 of a triangular number.*

2. Alternative naming:

   > *However, being in "one-to-one correspondence" is synonymous with being a bijection.*

3. Formulations and notations:

   > *The binomial distribution is therefore given by $P_p(n|N)$ =[FORMULA].*
   > *For a binary relation R, one often writes aRb to mean that (a, b) is in $R \times R$.*

4. Partial definition:

   A center X is the triangle centroid of its own pedal triangle iff it is the symmedian point.

   This sentence was annotated as non-definition, because it does not define the symmedian point.
5. Description:

   The cyclide is a quartic surface, and the lines of curvature on a cyclide are all straight lines or circular arcs.

Most misclassified definitions (false negatives) can be described by an atypical grammatical structure. Examples of such sentences can be seen below:

Once one countable set S is given, any other set which can be put into a one-to-one correspondence with S is also countable.
The word cissoid means "ivy-shaped".
A bijective map between two metric spaces that preserves distances,
i.e., d(f(x), f(y)) = d(x, y), where f is the map and d(a, b) is the distance function.

We propose to deal with some of the identified error sources as follows. Partial definitions can probably be discarded by applying part-of-speech tagging and pronouns detection. Coreference resolution (CR) can be used for identification of the referred mathematical entity in a text. Additionally, the partial definitions problem should be resolved by reduction of the DE task to multi-sentence DE. Formulations and notations can probably be discarded by measuring the ratio between mathematical symbolism and regular text in a sentence. Sentences providing alternative naming for mathematical objects can be discarded if we are able to detect the truth definition and then select it from multiple candidates. It can also be probably resolved with the help of such types of CR as split antecedents and coreferring noun phrases.

4.11. Discussion

As can be seen from all four experiments, ensemble outperforms individual models, despite latter being trained on more data. This outcome definitely supports the superiority of the ensemble approach for both domains.

It is worth noting that BERT did not perform well on our task. We explain it by the difference between the general domain of its training and our application domain of definitions and the lack of syntactic information in its input representation.

The scores of individual models approve again that syntactic information embedded into a sentence representation usually delivers better performance in both domains.

As our cross-domain evaluation results show, general definition domain and mathematical domain are quite different and, therefore, transfer cross-domain learning performs significantly worse than traditional single-domain learning.

5. Conclusions

In this paper, we introduce a new approach for DE, using ensemble from deep neural networks. Because it is a supervised approach, we adjust the class distribution of our datasets with oversampling. We evaluate this approach on datasets from general and mathematical domains. Our experiments on four datasets demonstrate superiority of ensemble voting over multiple state-of-the-art methods.

In the future, we intend to adapt our methodology for multi-sentence definition extraction.

Author Contributions: Conceptualization, N.V. and M.L.; methodology, N.V. and M.L.; software, N.V.; validation, N.V. and M.L.; formal analysis, investigation, resources, N.V. and M.L.; writing—original draft preparation, N.V. and M.L.; writing—review and editing, N.V. and M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The WFMALL dataset is freely available for download from https://github.com/NataliaVanetik1/wfmall, accessed on 1 September 2021.

Acknowledgments: The authors express their deep gratitude to Guy Shilon and Lior Reznik for their help with server configuration.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript, in the order of their appearance:

DE	Definition Extraction
NLP	Natural Language Processing
WCL	World-Class Lattice
DL	Deep Learning
CRF	Conditional Random Fields
CNN	Convolutional Neural Network
LSTM	Long Short-Term memory
BLSTM	Bidirectional LSTM
SOTA	State of the Art
NN	Neural Network
FT	fastText word vectors
LR	Logistic Regression
RF	Random Forest

References

1. Vanetik, N.; Litvak, M.; Shevchuk, S.; Reznik, L. Automated discovery of mathematical definitions in text. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 13–15 May 2020; pp. 2086–2094.
2. Xu, J.; Licuanan, A.; Weischedel, R.M. *TREC 2003 QA at BBN: Answering Definitional Questions*; TREC: Gaithersburg, MD, USA, 2003; pp. 98–106.
3. Klavans, J.L.; Muresan, S. Evaluation of the DEFINDER system for fully automatic glossary construction. In Proceedings of the AMIA Symposium, American Medical Informatics Association, Washington, DC, USA, 3–7 November 2001; p. 324.
4. Schuyler, P.L.; Hole, W.T.; Tuttle, M.S.; Sherertz, D.D. The UMLS Metathesaurus: Representing different views of biomedical concepts. *Bull. Med. Libr. Assoc.* **1993**, *81*, 217. [PubMed]
5. Malaisé, V.; Zweigenbaum, P.; Bachimont, B. Detecting semantic relations between terms in definitions. In Proceedings of CompuTerm 2004: 3rd International Workshop on Computational Terminology, Geneva, Switzerland, 29 August 2004.
6. Saggion, H.; Gaizauskas, R.J. Mining On-line Sources for Definition Knowledge. In Proceedings of the International FLAIRS Conference, Miami Beach, FL, USA, 12–14 May 2004; pp. 61–66.
7. Saggion, H. Identifying Definitions in Text Collections for Question Answering. In Proceedings of the International Conference on Language Resources and Evaluation, LREC, Lisbon, Portugal, 26–28 May 2004.
8. Storrer, A.; Wellinghoff, S. Automated detection and annotation of term definitions in German text corpora. In Proceedings of the International Conference on Language Resources and Evaluation, LREC, Genoa, Italy, 24–26 May 2006; Volume 2006.
9. Borg, C.; Rosner, M.; Pace, G. Evolutionary algorithms for definition extraction. In Proceedings of the 1st Workshop on Definition Extraction. Association for Computational Linguistics, Borovets, Bulgaria, 18 September 2009; pp. 26–32.
10. Fahmi, I.; Bouma, G. Learning to identify definitions using syntactic features. In Proceedings of the Workshop on Learning Structured Information in Natural Language Applications, Trento, Italy, 3 April 2006.
11. Westerhout, E.; Monachesi, P.; Westerhout, E. Combining pattern-based and machine learning methods to detect definitions for elearning purposes. In Proceedings of the RANLP 2007 Workshop "Natural Language Processing and Knowledge Representation for eLearning Environments", Borovets, Bulgaria, 27–29 September 2007.
12. Westerhout, E. Definition extraction using linguistic and structural features. In Proceedings of the 1st Workshop on Definition Extraction. Association for Computational Linguistics, Borovets, Bulgaria, 18 September 2009; pp. 61–67.
13. Navigli, R.; Velardi, P. Learning word-class lattices for definition and hypernym extraction. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010; pp. 1318–1327.
14. Reiplinger, M.; Schäfer, U.; Wolska, M. 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. Association for Computational Linguistics, Jeju Island, Korea, 10 July 2012; pp. 55–65.
15. Jin, Y.; Kan, M.Y.; Ng, J.P.; He, X. 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, Seattle, WA, USA, 18–21 October 2013; pp. 780–790.
16. Boella, G.; Di Caro, L. 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), Sofia, Bulgaria, 4–9 August 2013; Volume 2, pp. 532–537.

17. Anke, L.E.; Saggion, H.; Ronzano, F. Weakly supervised definition extraction. In Proceedings of the International Conference Recent Advances in Natural Language Processing, Hissar, Bulgaria, 5–11 September 2015; pp. 176–185.
18. Anke, L.E.; Saggion, H. Applying dependency relations to definition extraction. In Proceedings of the International Conference on Applications of Natural Language to Data Bases/Information Systems, Montpellier, France, 18–20 June 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 63–74.
19. Li, S.; Xu, B.; Chung, T.L. Definition Extraction with LSTM Recurrent Neural Networks. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 177–189.
20. Anke, L.E.; Schockaert, S. 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), New Orleans, LA, USA, 2–4 June 2018; Volume 2, pp. 378–385.
21. Avram, A.M.; Cercel, D.C.; Chiru, C. UPB at SemEval-2020 Task 6: Pretrained Language Models for Definition Extraction. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, Barcelona, Spain, 12–13 December 2020; pp. 737–745.
22. Xie, S.; Ma, J.; Yang, H.; Lianxin, J.; Yang, M.; Shen, J. UNIXLONG at SemEval-2020 Task 6: A Joint Model for Definition Extraction. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, Barcelona, Spain, 12–13 December 2020; pp. 730–736.
23. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
24. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. In Proceedings of the NAACL, New Orleans, LA, USA, 1–6 June 2018.
25. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Harrahs and Harveys, Lake Tahoe, CA, USA, 5–10 December 2013; pp. 3111–3119.
26. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. FastText Word Vectors. Available online: https://fasttext.cc/docs/en/crawl-vectors.html (accessed on 1 January 2018).
27. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. Python Interface to CoreNLP Using a Bidirectional Server-Client Interface. Available online: https://github.com/stanfordnlp/python-stanford-corenlp (accessed on 1 January 2019).
28. Chollet, F. Keras. Available online: https://keras.io (accessed on 1 January 2015).
29. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: tensorflow.org. (accessed on 1 January 2015).
30. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
31. Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I.H. The WEKA data mining software: An update. *ACM Sigkdd Explor. Newsl.* **2009**, *11*, 10–18. [CrossRef]
32. Navigli, R.; Velardi, P.; Ruiz-Martínez, J.M. WCL Definitions Dataset. Available online: http://lcl.uniroma1.it/wcl/ (accessed on 1 September 2020).
33. Navigli, R.; Velardi, P.; Ruiz-Martínez, J.M. An Annotated Dataset for Extracting Definitions and Hypernyms from the Web. In Proceedings of the International Conference on Language Resources and Evaluation, LREC, Valetta, Malta, 19–21 May 2010.
34. Spala, S.; Miller, N.A.; Yang, Y.; Dernoncourt, F.; Dockhorn, C. DEFT: A corpus for definition extraction in free-and semi-structured text. In Proceedings of the 13th Linguistic Annotation Workshop, Florence, Italy, 1 August 2019; pp. 124–131.
35. Jin, Y.; Kan, M.Y.; Ng, J.P.; He, X. W00 Definitions Dataset. Available online: https://bitbucket.org/luisespinosa/neural_de/src/afedc29cea14241fdc2fa3094b08d0d1b4c71cb5/data/W00_dataset/?at=master (accessed on 1 January 2013).
36. Bird, S.; Dale, R.; Dorr, B.J.; Gibson, B.R.; Joseph, M.T.; Kan, M.; Lee, D.; Powley, B.; Radev, D.R.; Tan, Y.F. 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, Marrakech, Morocco, 26 May–1 June 2008.
37. Vanetik, N.; Litvak, M.; Shevchuk, S.; Reznik, L. WFM Dataset of Mathematical Definitions. Available online: https://github.com/uplink007/FinalProject/tree/master/data/wolfram (accessed on 1 January 2019).
38. Weisstein, E. *Wolfram Mathworld*. Available online: https://www.wolframalpha.com/ (accessed on 1 January 2019).
39. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP natural language processing toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 23–25 June 2014; pp. 55–60.
40. Honnibal, M.; Johnson, M. An Improved Non-monotonic Transition System for Dependency Parsing. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics, Lisbon, Portugal, 17–21 September 2015; pp. 1373–1378.
41. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning Word Vectors for 157 Languages. In Proceedings of the International Conference on Language Resources and Evaluation LREC, Miyazaki, Japan, 7–12 May 2018.
42. Veyseh, A.; Dernoncourt, F.; Dou, D.; Nguyen, T. A joint model for definition extraction with syntactic connection and semantic consistency. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 9098–9105.