machine learning &
knowledge extraction

Article

Towards Robust Text Classification with Semantics-Aware Recurrent Neural Architecture

Blaž Škrlj [1,2], Jan Kralj [1], Nada Lavrač [1,3] and Senja Pollak [1,4,*]

[1] Jožef Stefan Institute, 1000 Ljubljana, Slovenia; blaz.skrlj@ijs.si (B.Š.); jan.kralj@ijs.si (J.K.); nada.lavrac@ijs.si (N.L.)
[2] Jožef Stefan International Postgraduate School, 1000 Ljubljana, Slovenia
[3] School of Engineering and Management, University of Nova Gorica, 5000 Nova Gorica, Slovenia
[4] Usher Institute, Medical School, University of Edinburgh, Edinburgh EH16 4UX, UK
* Correspondence: senja.pollak@ijs.si

Received: 21 February 2019; Accepted: 1 April 2019; Published: 4 April 2019

Abstract: Deep neural networks are becoming ubiquitous in text mining and natural language processing, but semantic resources, such as taxonomies and ontologies, are yet to be fully exploited in a deep learning setting. This paper presents an efficient semantic text mining approach, which converts semantic information related to a given set of documents into a set of novel features that are used for learning. The proposed Semantics-aware Recurrent deep Neural Architecture (SRNA) enables the system to learn simultaneously from the semantic vectors and from the raw text documents. We test the effectiveness of the approach on three text classification tasks: news topic categorization, sentiment analysis and gender profiling. The experiments show that the proposed approach outperforms the approach without semantic knowledge, with highest accuracy gain (up to 10%) achieved on short document fragments.

Keywords: recurrent neural networks; text mining; semantic data mining; taxonomies; document classification

1. Introduction

The task of classifying data instances has been addressed in data mining, machine learning, database, and information retrieval research [1]. In text mining, document classification refers to the task of classifying a given text document into one or more categories based on its content [2]. A text classifier is given a set of labeled documents as input, and is expected to learn to associate the patterns appearing in the documents to the document labels. Lately, deep learning approaches have become a standard in natural language-related learning tasks, showing high performance in different classification tasks involving various text types, including sentiment analysis of tweets [3] and news categorization [4].

Semantic data mining denotes a data mining approach where (domain) ontologies are used as background knowledge in the data mining process [5]. Semantic data mining approaches have been successfully applied in semantic subgroup discovery [6], data visualization [7], as well as text classification [8,9]. Provision of semantic information allows the learner to use features on a higher semantic level, allowing for data generalization. The semantic information is commonly represented as relational data in the form of networks or ontologies. Even though there are many sources of such knowledge, approaches capable of leveraging such information in a deep learning setting are still scarce.

This paper proposes a novel approach where semantic information in the form of taxonomies (i.e., ontologies with only hierarchical relations) is propositionalized and then used in a recurrent neural network architecture. The proposed SRNA (Semantics-aware Recurrent Neural Architecture) approach has been tested on a document classification task, while special attention is paid to the robustness of the method on short document fragments. Classification of short or incomplete documents is useful in a large variety of tasks. For example, in author profiling, the task is to recognize author's characteristics, such as age or gender [10], based on a collection of author's text samples, where the effect of data size is known to be an important factor influencing classification performance [11]. A frequent text type for this task are tweets, where a collection of tweets from the same author is considered a single document, to which a label must be assigned. The fewer instances (tweets) we need, the more powerful and useful is the approach. In a similar way, this holds true for nearly any kind of text classification task. For example, for labeling a news article with a topic tag, using only snippets or titles and not the entire news, may be preferred due to limited text availability or required processing speed.

It has been demonstrated that deep neural networks need a large amount of information in order to learn complex representations from text documents, and that state-of-the-art models do not perform well when incomplete information is used as input [12]. This work addresses an open problem of increasing the robustness of deep neural network-based classifiers in such settings by exploring to what extent the documents can be truncated without affecting the learner's performance.

This work is structured as follows. Section 2 presents the background and related work. Section 3 introduces the proposed SRNA architecture, where semantic information in the form of taxonomies is propositionalized and used in a recurrent neural architecture. Sections 4 and 5 present the experimental setup and results of the evaluation on three publicly available data sets, with a special focus on how the constructed semantic vectors affect the classifier's performance. We conclude the paper in Section 6 with the plans for further work.

2. Background and Related Work

This section outlines the background and the related work in semantics-aware data mining and deep learning architectures.

2.1. Document Representation and Semantic Context

Document classification is highly dependent on *document representation*. In simple bag-of-words representations, the frequency (or a similar weight such as term frequency inverse document frequency) of each word or *n*-gram is considered as a separate feature. More advanced representations group words with similar meaning together. The approaches include Latent Semantic Analysis (LSA) [13], Latent Dirichlet Allocation (LDA) [14], and more recently word embeddings [15], which transform data instances (documents) into feature vectors in a lower-dimensional numeric vector space. One of the well known algorithms for word embedding is word2vec [15], which uses a two-layer shallow neural network architecture to capture the word context of the given text. As word2vec captures limited contextual information, recently introduced embedding approaches such as GloVe [16] and FastText [17] attempt to address these issues. Individual embeddings (feature vectors) are positioned closer if they are contextually more similar. Both embedding and LSA-based approaches have significantly improved in the recent years, both in terms of scalability, as well as in terms of their predictive power [18,19].

It has been previously demonstrated that context-aware algorithms significantly outperform the naive learning ones [20]. Neural networks can learn word representations by using their context, and are as such especially useful for text classification tasks. We refer to such semantic context as the *first-level context*.

Second-level context can be introduced by incorporating extensive amounts of *background knowledge* (e.g., in the form of ontologies or taxonomies) into a learning task, which can lead to improved performance of semantics-aware rule learning [6], subgroup discovery [21], and random forest learning [22]. In text mining, Elhadad et al. [23] report an ontology-based web document classifier, while Kaur et al. [24] propose a clustering-based algorithm for document classification, which also benefits from the knowledge stored in the underlying ontologies.

Cagliero and Garza [20] report a custom classification algorithm, which can leverage taxonomies, and demonstrate—on a case study of geospatial data—that such information can be used to improve classification. Use of hypernym-based features for classification tasks has been considered previously. The Ripper rule learner was used with hypernym-based features [8], while the impact of WordNet-based features for text classification was also evaluated [9], demonstrating that hypernym-based features significantly impact the classifier performance.

Even though including background information in deep learning has yet to be fully exploited, there are already some *semantic deep learning* approaches available for text classification. Tang et al. [19] have demonstrated that word embedding approaches can take into account semantics-specific information to improve classification. Ristoski et al. [25] show that embedding-based approaches are useful for taxonomy induction and completion. Liu et al. [26] address incorporation of taxonomy-derived background knowledge as a constrained optimization problem, demonstrating that semantic information can be valuable for the tasks of entity recognition and sentence completion. Finally, Bian et al. [27] leverage morphological, syntactic, and semantic knowledge to achieve high-quality word embeddings and prove that knowledge-powered deep learning can enhance their effectiveness.

2.2. Deep Learning Architectures

This section introduces *deep learning architectures* for text classification.

A two-layer neural network has been introduced as part of the word2vec embedding approach [15]. Recently, deeper architectures have proven to work well in document classification tasks [28–30], where a neural network is given a set of vectors, whose elements are e.g., individual word indexes that are directly used to produce class predictions. These approaches include *convolutional neural networks*, which have been previously proven to work well for image classification [31,32]. A convolution is defined as:

$$s(t) = (x * w)(t) = \sum_{m=-\infty}^{\infty} x(m)w(t-m),$$

where x is the input function, m the input vector dimensionality and w is a kernel.

Kernels are smaller sub-matrices, which are applied in the process of convolution, and result in a modified origin matrix that can represent e.g., an image or a text sequence.

A convolutional neural network consists of at least three different types of computational layers: a convolution layer, a pooling layer, and a dense fully connected layer. The convolution layer returns convolutions computed on the given (single or multidimensional) inputs. Such a layer is normally followed by a pooling layer. Here, sets of neurons' outputs are merged into a single real number r. Common pooling layers include maximum and average pooling. Finally, the fully connected layer consists of a set of neurons, such that each neuron in the fully connected layer is connected with each neuron in the previous layer. In most contemporary convolutional architectures, fully connected layers (the first types of layers to be used in neural networks) are only used in the final stages due to their prohibitive computational cost. Single-dimensional convolutional networks are used extensively in natural language processing (NLP) tasks [28,33]. In a standard setting, vectors of word indexes are used as input for a deep learning-based text classifier. The first layer in such architectures is responsible for the construction of a lower-dimensional word index embedding, which is further used for learning. The objective of this layer is to project the high dimensional input into a lower dimensional vector space, more suitable for computationally expensive learning [34].

Recently, *recurrent neural networks* have gained significant momentum [35]. A recurrent neural network is a type of architecture with recurrent connections between individual neurons. Similarly to feedback loops in biology, such architectures to some extent enable memory storage. The most commonly used recurrent architecture for sequence classification include the so-called Long-Short Term Memory (LSTM) cells [36] and Gated Recurrent Units (GRUs) [37].

A single LSTM cell consists of three main gates: the input, output and the forget gate (see Figure 1). Individual activations within a LSTM cell are defined as sigmoid functions:

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

All three gates together form a feedback loop preserving gradients during the training. The main benefit for sequence learning is that LSTMs to some extent solve the vanishing gradient problem, i.e., long term signals remain in the memory, whereas a simple feedforward architecture is prone to vanishing gradients.

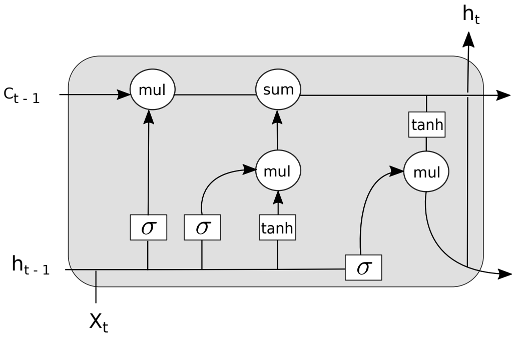

Figure 1. The LSTM cell. The forget gate is responsible for selective information filtering during the learning step [36,38]. Here, the C_{t-1} corresponds to the memory state at learning step $t-1$. We refer the interested reader to [38] for a more detailed overview of the LSTM cells shown here.

One issue common to all neural network models is that they often overfit the data. One of the most common solutions is the introduction of dropout layers [39] (at each training step, a percentage of neurons is omitted from being trained). We use them for regularization.

To achieve the state-of-the-art performance, sets of trained neural networks can be combined into neural ensembles. Some of the well known approaches which exploit this property include HDLTex [40] and RMDL [38]. Both approaches focus on learning of different aspects of the data set, yielding robust and powerful ensamble classification methods for e.g., text classification.

Large success of neural networks for classification is due to their capability of learning latent relationships in the data. In this work, we evaluate how additional information in the form of taxonomies affects the learning process. Even though feature engineering is becoming less relevant in the era of deep learning [41], we believe that integrating background knowledge can potentially improve classification models, especially when data is scarce, which is one of the currently unsolved problems related to deep architectures.

3. Proposed SRNA Approach

This section presents the proposed SRNA (Semantics-aware Recurrent Neural Architecture) approach, which leverages knowledge from taxomomies for construction of novel features for use in a custom deep neural network architecture. Figure 2 outlines the proposed two-step approach. In step 1 (described in Section 3.1), an input corpus $\mathcal{D}$ and a hypernym taxonomy are used to construct separate feature matrices D and S. In step 2 (described in Section 3.2), the two matrices are input into a hybrid neural network architecture to predict labels of new input documents.

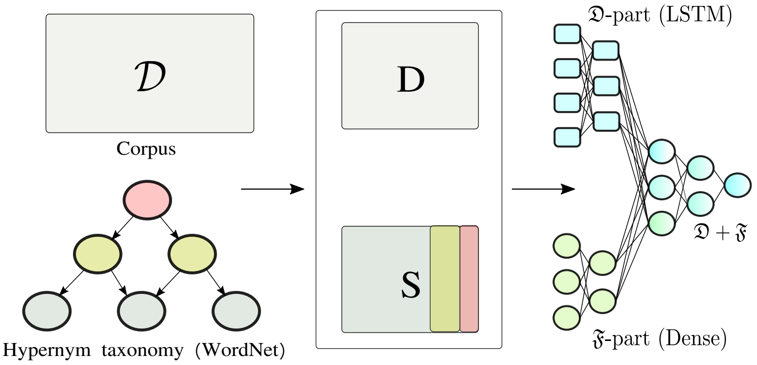
$\mathcal{D}$
Corpus
Hypernym taxonomy (WordNet)
D
S
$\mathfrak{D}$-part (LSTM)
$\mathfrak{D} + \mathfrak{F}$
$\mathfrak{F}$-part (Dense)

Figure 2. Visualization of the SRNA approach to semantic space propositionalization and learning. Left: A document corpus $\mathcal{D}$ and a hypernym taxonomy (WordNet). Middle: A matrix of word indexes D obtained from corpus $\mathcal{D}$, and a matrix of semantic features vectors S (with the same number of rows as D), with features obtained from different levels of the taxonomy. Right: A hybrid neural network architecture is learned from the word index vectors and the semantic feature vectors. Note that sequential word information is present only in the vectors constituting matrix D (word indices), hence part of the architecture exploits sequential information, whereas the constructed semantic features are input to the dense feedforward part of the architecture. Prior to the final layer, intermediary layers of both parts of the network are merged.

3.1. Propositionalization of the Semantic Space

The first step of the SRNA approach is hypernym identification and selection. We investigate how hypernyms can be used as additional background knowledge to possibly improve the classification. We rely on WordNet [42], a large and widely used lexical resource, in which words are annotated with word senses (i.e., word meanings) and connected by semantic relations, including synonymy (e.g., *car* ↔ *auto*), hypernymy (e.g., *car* → *vehicle*) and hyponymy (e.g., *vehicle* → *car*). In this work, we explore only the space of hypernymy relations. The obtained hierarchical structure is thus a taxonomy. In order to leverage the extensive knowledge stored in word taxonomies, a propositionalization algorithm was developed, performing the fusion of the original set of documents $\mathcal{D}$, represented by word index matrix D of dimension $N \times \ell$ (is the user defined parameter for determining the dimension of their feature vectors, corresponding to the number of word indices used), with newly constructed semantic features. These features are the hypernyms, forming the columns of the semantic feature matrix S of dimension $N \times m$. The process of propositionalization merges (concatenates) the original matrix D and the sematic feature matrix S into novel matrix DS of dimension $N \times (\ell + m)$.

The semantic feature matrix S is constructed as follows. First, the corpus is processed document by document. For each document d, we collect the words appearing in d and, for every word w, we store the number of times it appears (its "frequency"). Next, for every w in d, we obtain the set of its *representative hypernyms*. We make no attempt at word-sense disambiguation and leave this aspect for further work. Instead, for words with several corresponding *synsets* (words with multiple senses), a hypernym h is representative if it is a hypernym for every sense of the word w, by which we avoid the fact that we are missing information on the actual sense of the word in context. Thus, we identify the set of all corresponding WordNet synsets of w (denoted by $\mathfrak{S}_w$), and the representative hypernyms of word w, denoted by $\mathfrak{A}_w$, are hypernyms of all the synonyms in $\mathfrak{S}_w$:

$$\mathfrak{A}_w = \bigcap_{s \in \mathfrak{S}_w} \{h | h \text{ is a hypernym of } s\}.$$

We also store "frequencies" of all representative hypernym counts—for a hypernym h, the frequency of h is defined as the sum of the frequencies of all of its hyponyms. Note that more general hypernyms will occur more often, hence the hierarchical relations between hypernyms are captured via hypernym frequency.

Once representative hypernyms are identified for all words appearing in a document d, the set $\mathfrak{H}_d$ is constructed as $\mathfrak{H}_d = \bigcup_{w \in d} \mathfrak{A}_w$, and, once this set is constructed for all documents, the set $\mathfrak{H}$ is constructed as the set of all representative hypernyms of the corpus, i.e., $\mathfrak{H} = \bigcup_{d \in \mathcal{D}} \mathfrak{H}_d$. Throughout this process, counts of hypernym occurences are stored for each document, and once all documents are processed, features are constructed based on the overall hypernym counts. The number of semantic feature vectors to be constructed, denoted λ, is a parameter of the proposed algorithm. The upper bound for λ is $|\mathfrak{H}|$, i.e., the number of all representative hypernyms. We propose three approaches, which prioritize the hypernyms according to their frequency of occurrence. The three approaches used to select λ hypernyms for semantic feature vector construction are:

- top λ most frequent terms,
- last λ terms (very rare terms),
- a set of random λ terms.

The obtained matrix can be used for learning either as a separate semantic feature set (S) or as the whole DS matrix along with word-index matrix D.

3.2. Learning from the Semantic Space

The second step of the SRNA approach consists of training a deep architecture using the expanded feature matrix (DS) obtained in the first step. In SRNA, semantic features are fed into a deep architecture along with document vectors. The outline of the architecture, shown in Figure 2, can be represented in three main parts. The first part is responsible for learning from document vectors, and is denoted by $\mathfrak{D}$. The second part learns from the constructed semantic vectors, denoted as $\mathfrak{S}$. Finally, before output layer, outputs of $\mathfrak{D}$ and $\mathfrak{S}$ are merged and processed jointly. We denote this part by $(\mathfrak{D} + \mathfrak{S})$. We give exact (hyperparameter) parameterization of the architecture in Section 4.

The recurrent part of the network, represented by the $\mathfrak{D}$ part, is in this work defined as follows. An input vector of word indices is first fed into an embedding layer with dropout regularization. The resulting output is used in a standard LSTM layer. The output of this step is activated by a ReLU activation function, defined as:

$$ReLU(x) = max(0, x).$$

The output of this layer is followed by a MaxPooling layer. Here, maximal values of a kernel moving across the input vector are extracted. Finally, a dense layer with dropout regularization is used. Formally, the $\mathfrak{D}$ part of the network can be defined as:

$$\begin{aligned} L_{(1)} &= Dropout(Emb(D)), \\ L_{(2)} &= MaxPooling(ReLU_{(2)}(LSTM(L_{(1)}))), \\ L_{(3w)} &= Dropout(W_{(3)}^T L_{(2)} + b_{(3)}). \end{aligned}$$

The $\mathfrak{S}$ part of the architecture similarly consists of fully connected layers. The input for this part of the network are generated semantic features S. It can be represented as:

$$\begin{aligned} L_{(1)} &= Elu_{(1)}(W_{(1)}^T S + b_{(1)}), \\ L_{(2)} &= Dropout(L_{(1)}), \\ L_{(3s)} &= Elu_{(3)}(W_{(3)}^T L_{(2)} + b_{(3)}). \end{aligned}$$

Here, we use the exponential linear unit [43], defined as

$$Elu(x) = \begin{cases} x, & \text{for } x \geq 0, \\ c(e^x - 1), & \text{for } x < 0. \end{cases}$$

Here, c is a constant determined during parameterization of the architecture. Outputs of D and S parts of the architecture are concatenated and used as input to a set of fully connected (dense) layers (M), defined as:

$$\begin{aligned} L_{(1)} &= concat(L_{(3w)}, L_{(3s)}), \\ L_{(2)} &= Elu(Dropout(W_{(2)}^T L_{(1)} + b_{(2)})), \\ L_{(3f)} &= \sigma(W_{(3)}^T L_{(2)} + b_{(3)}). \end{aligned}$$

The *concat* operator merges the outputs of the two individual parts of the network into a single matrix. For concatenation, one of the dimensions (in our case, N, the number of instances) of the two output layers must be the same.

Finally, the output layer $L_{(3f)}$ includes one neuron for each class in the data set. We use binary cross entropy as the loss function. The exact layer parameterizations are discussed in the experimental setting section. The Adam optimizer [44] was chosen due to faster convergence. Formulation of the whole SRNA approach is presented in Algorithm 1.

Algorithm 1 Semantic space propositionalization with learning.

```
Data: corpus $\mathcal{D}$,WordNet taxonomy
for all document in $\mathcal{D}$ do
    for all word in document do
        Find hypernyms (based on WordNet) for word, store them and their counts
    end for
    Compute intersection of hypernym paths
end for
Assign feature values based on hypernym frequency in a document
$S$ := Select top $\lambda$ hypernyms as features based on overall hypernym frequency
$D$ := transform $\mathcal{D}$ into a matrix of word indices Learn a deep model using matrices $D$ and $S$.
```

The proposed algorithm's temporal complexity is linear with respect to document number, making it scalable even for larger corpora. Similarly, the frequency count estimation is not computationally expensive. One of the key goals of this work was to explore how semantic information, derived from individual documents, affects the learner's performance. The SRNA code is accessible at https://gitlab.com/skblaz/srna.

In the next section, we continue with the experimental setting where we evaluate the proposed methodology.

4. Experimental Setting

We compared the performance of the SRNA approach against multiple baseline classifiers. We tested the methods on three benchmark data sets. We next describe the experimental setting in more detail.

4.1. Data Sets

All documents were padded to the maximum dimension of 150 words. We conduct a series of experiments, where we truncate the training documents (D) to lengths from 15 to 150 by the increment of 10. The semantic feature matrix S is constructed using truncated documents. Note that the number of documents remains the same; we only experiment with the number of words per document. The results were obtained using 10 fold stratified cross validation. We tested the proposed approach on three data sets, listed below.

Reuters data set consists of 11,263 newspaper articles, belonging to 46 different topics (classes). This data set is loaded via the Keras library, where it is also publicly accessible (https://keras.io/datasets/).

IMDB review data set consists of 50,000 reviews. Here, the goal is to predict the sentiment of individual reviews (positive or negative). The data set was obtained from the Keras library [45], where it is also accessible.

PAN reviews data set consists of reviews written by 4160 authors (2080 male and 2080 female). Reviews written by the same author are concatenated in a single document. The goal is to classify the author's gender. Detailed description of the data set is given in [10].

4.2. Semantic Feature Construction

We generated 1000 semantic features for each of the feature selection approaches. After initial tests, we observed that the sparse feature set (rarest hypernyms) outperforms the other two approaches, thus this setting was used for further tests. To reduce the number of candidate hypernym features, we introduce a minimum frequency threshold—a threshold above which we consider a hypernym as a potential feature. The frequency threshold used was 10, i.e., a hypernym is common to at least 10 words from the corpus in order to be considered for feature construction. (Note that this step of the approach could be possibly improved using e.g., the RelieF) [46] branch of algorithms.

4.3. Deep Neural Architectures Used

As part of experimental evaluation, we test three deep learning models, two with inclusion of semantic vectors and a baseline ConvNet. All the models are initiated in the same way.

SRNA: Recurrent architecture. This is the proposed architecture that we described in Section 3. It learns by using LSTM cells on the sequential word indices, and simultaneously captures semantic meaning using dense layers over the semantic feature space.

Baseline RNN. The baseline RNN architecture consists of the non-semantic part of SRNA. Here, a simple unidirectional RNN is trained directly on the input texts.

Baseline CNN. The baseline neural networks used are a 1D convolutional neural network and a recurrent neural network with the same architecture as SRNA, where we omit the semantic part. Here, only word index vectors are used as inputs. The network was parameterized as follows. The number of filters was set to 64, the kernel size used was 5. The MaxPooling region was of size 5. The outputs of the pooling region were used as input to a dense layer with 48 neurons, followed by the final layer.

One of the main problems with small data sets and neural networks is overfitting. Each neural network is trained incrementally, where the training is stopped as soon as the network's performance starts to degrade. Furthermore, dropout layers are used for additional regularization (the dropout rate was set to 0.5). The alpha parameter of each Elu activation function was set to 1.

As an additional baseline, we implemented also two non-neural classifiers, i.e., the random forest classifier, and a support vector machine, where we also tested how semantic vectors contribute to classification accuracy.

The random forest (RF) classifier was initialized as follows: number of trees for classification from documents was set to the average document length present in a given corpus rounded to the closest integer. One versus all (OVA) classification scheme was used for the multi-class Reuters task. To evaluate the semantic addition, we implemented two variants of random forests, both learned from identical input as given to neural networks. **Semantic RF** is the random forest that leverages semantic information (i.e., $D + S$ matrix), while **RF** is trained exclusively on TF-IDF word vectors obtained from D.

Support vector machine (SVM) classifier [47] was trained as follows. We used the RBF kernel and the C value determined over a grid search over range [0.1,1,10]. Similarly to random forests, we also implemented the version called **Semantic SVM**, which uses SRNA's semantic features along with TF-iDF matrix as input.

Other Technical Details

The SRNA approach was along with Baseline RNN and CNN architectures implemented in Keras framework, where we used the Tensorflow computational back-end [48]. The other classifiers were called from the Scikit-learn Python library [49]. All approaches were tested on a Nvidia Titan GPU (NVIDIA, Santa Clara, CA, USA). The baseline Random Forest classifier was implemented in Scikit-learn [49]. Matrix-based operations in the propositionalization step used the Numpy library [50].

5. Results and Discussion

For all data sets, we measure the accuracy. In case of Reuters, which is a multiclass problem, the exact accuracy is also termed subset accuracy (or exact match ratio). We also compute the F1 score for the IMDB and PAN data sets, and micro F1 for Reuters. Each experiment with 10 fold cross validation is repeated five times, and the results are averaged. To statistically evaluate the results, we used the Friedman's test, followed by the Nemenyi post hoc correction. The results are presented according to the classifier's average ranks along a horizontal line [51]. The obtained critical distance diagrams are interpreted as follows: if one or more classifiers are connected with a bold line, their performance does not differ significantly (at alpha = 0.05). We rank the classifiers for each data set, for each individual subsample. Furthermore, we visualize the performance of SRNA compared to baseline RNN using the recently introduced Bayesian hierarchical *t*-test—a Bayesian alternative to pairwise classifier comparison over multiple data sets [52]. Here, instead of significance level, a rope parameter is set. This parameter determines the threshold, under which we consider the difference in classifier performance to be the same. In this work, we set this threshold to 0.01. Note that the hierarchical Bayesian *t*-test offers the opportunity to explore the pairwise comparison of classifiers in more detail, hence we use it to inspect the SRNA vs. Baseline RNN combination.

For different document lengths, we calculate the accuracy and F1 scores, for which the plots (for the sequence length up to 100) are provided in Figures 3 and 4, respectively. It can be seen that, on the Reuters data set, SRNA outperforms other approaches in terms of Accuracy and F1, while for the other two data sets it achieves comparable results to baseline RNN and CNN.

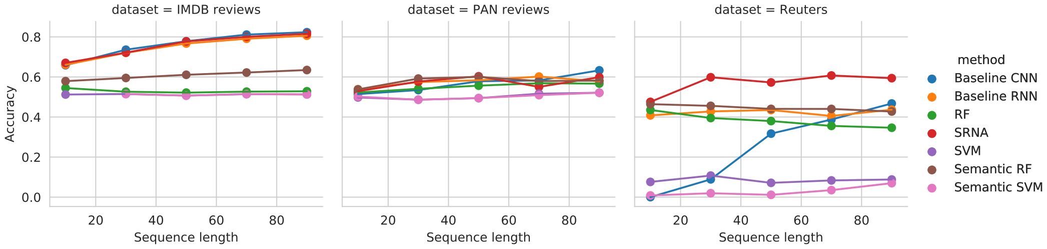

Figure 3. Accuracy results on three benchmark data sets.

We also present critical distance diagrams for the accuracy (Figure 5) and F1 measures (Figure 6). From the ranks, we can see that the SRNA approach outperforms all other baselines. However, the differences in performance between the SRNA approach and Baseline RNNs (as well as most of other classifiers) are not significant, and are data set dependent.

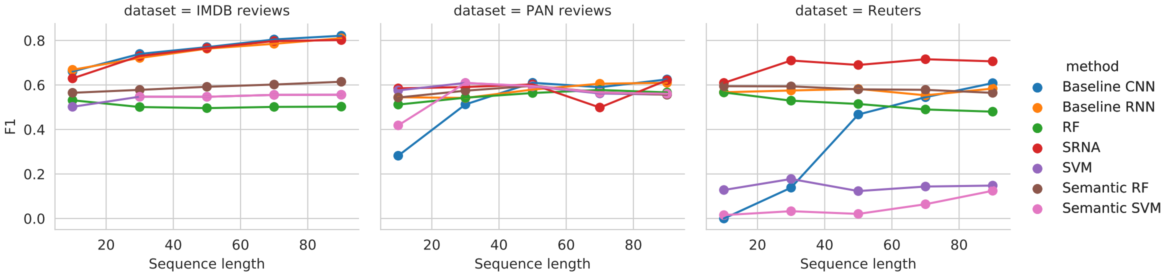

Figure 4. F1 results on three benchmark data sets.

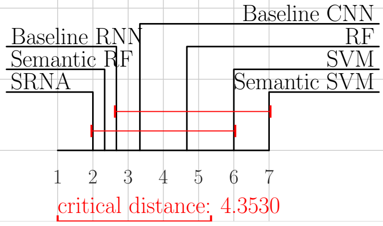

Figure 5. Accuracy—CD diagram.

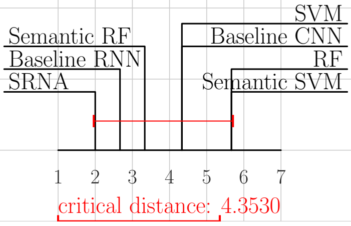

Figure 6. (Micro) F1—CD diagram.

Interestingly, the semantic feature-augmented random forests on average outperform their basic counterparts. This observation indicates that the semantic features could be used in a general classification setting, where an arbitrary classifier could benefit from the background knowledge introduced. Rigorous, large-scale experimental proof of this statement is out of the scope of this study.

As the goal of the proposed SRNA approach is to improve learning on very small data sets, with very limited data, we further investigate the classifier's performance on up to 100 words (see Figures 3 and 4).

When the considered recurrent architectures were inspected in more detail (Figure 7), we can observe that there is a higher probability that SRNA outperforms (Prob = 0.64) the baseline RNNs (Prob = 0.30), when the region of practical equivalence (ROPE) is set to 0.01, even though the performances of the two architectures are very similar. As an input to this test, we used differences in classifiers' performances from five repetitions of 10 fold cross validation.

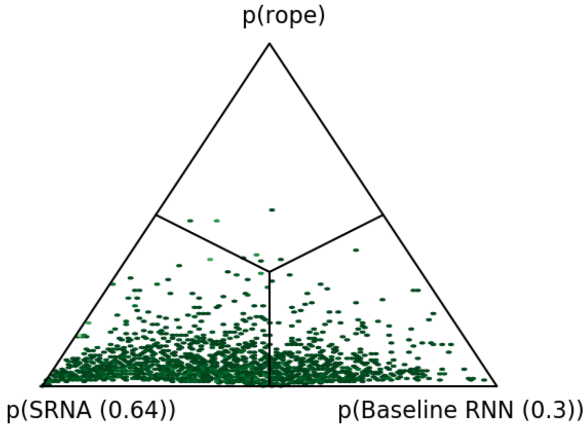

Figure 7. Sampled probability density of differences in classifier performance. Overall, the SRNA approach outperforms the baseline RNN, yet the larger differences in performance (e.g., Reuters data set) are data set-dependent. Higher probability of winning (0.64) in favour of SRNA indicates that semantic features potentially improve performance. Note that the ROPE parameter was for this test set to 0.01.

We further investigate the reasons why the baseline convolutional network performs very poorly when only up to 50 words are used. We believe the poor performance is related to the small data size. The CNN learns normally on very reduced documents, yet when its predictions were inspected, we observed it was not able to produce a single positive classification.

This behaviour was observed for document length ≤ 50, which resulted in two valid classifications ($length = 50$), whereas all other classifications ($length < 50$) returned ∼0% accuracy. The difference in accuracy for very short document lengths serves as an additional empirical proof that semantic vectors can at least augment the signal up to the classification threshold when using the SRNA.

The SVM approaches do not perform well in the conducted experiments. We believe the reason for this could lie in too small grid search region, as well as the noise potential introduced by semantic features. This indicates that the semantic features could be further pruned—such noise can have observable effects on the network's performance when semantic vectors are merged with the word vectors.

In addition, we observe that the SVM classifier did not perform well also when semantic features were added. Even though we did not test the regularization (C) range exhaustively, we believe that the SVMs' performance could be further improved. Moreover, the RBF kernel is not necessarily the optimal kernel choice.

Furthermore, we discuss the performance of random forests. The random forest classifier is in the majority of settings outperformed by other approaches (apart from SVMs), which is not surprising as very simple forest construction was used. However, we can see that with random forests the use of semantic features provides improvement. As compared to SVMs, random forests use a relatively low number of features; it is therefore easier to observe a difference in performance when novel features are introduced.

Interestingly, the random forest's performance appears to degrade in the case of the Reuters data set, which could indicate overfitting. As we used an OVA classification scheme, this decline in performance could be possibly solved by more advanced multi-class approaches, such as some form of predictive clustering trees. It is also possible that the problem is simply too hard for a random forest classifier used in this study, as it was not able to recognize any meaningful pattern, useful for classification into one of the possible topics.

Even though this study is not devoted to improving the overall state-of-the-art classification performance (SOTA), but to demonstrate how semantic features contribute to their semantically unaware counterparts, and especially how semantic features can be introduced in the neural architectures, we briefly discuss here SOTA results.

Currently, the best accuracy for the IMDB data set is estimated at around 98% for an approach that is based on paragraph vectors [53]. The authors compared their approach also with simple LSTMs (as used for baseline in this study), and obtained accuracies of 96%. We tested our baseline on the whole data set, and it performed similarly (95.3%), which serves as a validation of the baseline approach used in this study. Next, the accuracy on the Reuters data set was recently reported to be 80–85%, where multi-objective label encoders were used [54]. Our baseline implementation performs with 75% accuracy. Finally, SOTA for gender classification on PAN 2014 was reported to be around 73% [10].

Even if we investigated a particular aspect of text classification, not directly associated with SOTA, we will try to perform a more systematic evaluation to SOTA approaches in future work, however there are some limitations, such as computational cost of training very large networks and the fact that the majority of SOTA approaches do not account for a situation with sparse data. However, we believe that the proposed approach can be adapted to make current SOTA architectures more robust, especially when only fragments of inputs are considered.

6. Conclusions and Further Work

We developed an approach for propositionalization of semantic space in the form of taxonomies to improve text classification tasks. We explore possible deep architectures, which learn separately from the two feature spaces and prove that construction of such architectures can significantly improve overall classification on short document fragments. As we tested only three simple approaches for feature selection, this work could further benefit from more advanced feature selection techniques, such as the ones based on evolutionary computation or ReliefF branch of algorithms. We believe a more sophisticated feature selection approach would result in more relevant features, and could as such significantly speed up the learning phase. Furthermore, the approach could be tested in a setting where no feature selection is performed at all—for such experiments, one would need significantly more performant GPUs than the ones used in this experiment. We believe the neural networks would be able to select relevant features in an end-to-end manner.

As the results in this study indicate, recurrent neural architecture can indeed benefit from addition of semantic information, and part of the further work includes more extensive experimental tests, where state-of-the-art approaches, such as RMDL, HDLTex or hierarchical attention networks shall be combined with the proposed hypernym features.

As current state-of-the-art text classification approaches also work on the character level, it is yet to be investigated whether the proposed approach can also boost performance for character level architectures. Furthermore, the SRNA approach could potentially benefit from different types of recurrent layers, such as, for example, gated recurrent units (GRUs).

Last but not least, in a higher performance setting, the effects of semantic features could be evaluated on current SOTA algorithms, as well as on inherently short texts, such as tweets and comments. We will also include comparison of the proposed approach of semantic knowledge integration to enrichment with precomputed word embeddings.

Author Contributions: Conceptualization, B.Š., S.P. and J.K.; methodology, B.Š., J.K.; software, B.Š.; validation, B.Š., J.K., N.L. and S.P.; formal analysis, J.K.; investigation, S.P., B.Š.; resources, B.Š., S.P.; data curation, B.Š.; writing—original draft preparation, all authors; writing—review and editing, all authors; visualization, B.Š.; supervision, N.L., S.P.; project administration, S.P., N.L.; funding acquisition, S.P., N.L.

Funding: The work of the first author was funded by the Slovenian Research Agency through a young researcher grant. The work of other authors was supported by the Slovenian Research Agency (ARRS) core research programme *Knowledge Technologies* (P2-0103) and ARRS funded research project *Semantic Data Mining for Linked Open Data* (financed under the ERC Complementary Scheme, N2-0078). This paper is supported also by the European Union's Horizon 2020 research and innovation programme under Grant No. 825153, EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media). The results of this publication reflect only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

Acknowledgments: The GPU used for this research was donated by the NVIDIA Corporation.

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

References

1. Aggarwal, C.C.; Zhai, C. A survey of text classification algorithms. In *Mining Text Data*; Springer: Boston, MA, USA, 2012; pp. 163–222.
2. Sebastiani, F. Machine learning in automated text categorization. *ACM Comput. Surv.* **2002**, *34*, 1–47. [CrossRef]
3. Tang, D.; Qin, B.; Liu, T. Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1422–1432.
4. Kusner, M.; Sun, Y.; Kolkin, N.; Weinberger, K. From word embeddings to document distances. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 957–966.
5. Ławrynowicz, A. *Semantic Data Mining: An Ontology-Based Approach*; IOS Press: Amsterdam, The Netherlands, 2017; Volume 29.
6. Vavpetič, A.; Lavrač, N. Semantic subgroup discovery systems and workflows in the SDM toolkit. *Comput. J.* **2013**, *56*, 304–320. [CrossRef]
7. Adhikari, P.R.; Vavpetič, A.; Kralj, J.; Lavrač, N.; Hollmén, J. Explaining mixture models through semantic pattern mining and banded matrix visualization. *Mach. Learn.* **2016**, *105*, 3–39. [CrossRef]
8. Scott, S.; Matwin, S. Text classification using WordNet hypernyms. In Proceedings of the Workshop on Usage of WordNet in Natural Language Processing Systems, Montreal, QC, Canada, 16 August 1998; University of Montreal: Montreal, QC, Canada, 1998; pp. 45–51.
9. Mansuy, T.N.; Hilderman, R.J. Evaluating WordNet features in text classification models. In Proceedings of the FLAIRS Conference, Melbourne Beach, FL, USA, 11–13 May 2006; American Association for Artificial Intelligence: Menlo Park, CA, USA, 2006; pp. 568–573.
10. Rangel, F.; Rosso, P.; Chugur, I.; Potthast, M.; Trenkmann, M.; Stein, B.; Verhoeven, B.; Daelemans, W. Overview of the 2nd author profiling task at PAN 2014. In Proceedings of the Working Notes Papers of the CLEF Conference, Sheffield, UK, 15–18 September 2014; pp. 1–30.
11. Rangel, F.; Rosso, P.; Verhoeven, B.; Daelemans, W.; Potthast, M.; Stein, B. Overview of the 4th author profiling task at PAN 2016: Cross-genre evaluations. In Proceedings of the Working Notes Papers of the CLEF Conference, Evora, Portugal, 5–8 September 2016; pp. 750–784.
12. Cho, J.; Lee, K.; Shin, E.; Choy, G.; Do, S. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? *arXiv* **2015**, arXiv:1511.06348.
13. Landauer, T.K. *Latent Semantic Analysis*; Wiley Online Library: Hoboken, NJ, USA, 2006.
14. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
15. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
16. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
17. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *arXiv* **2016**, arXiv:1607.04606.
18. Song, G.; Ye, Y.; Du, X.; Huang, X.; Bie, S. Short text classification: A survey. *J. Multimed.* **2014**, *9*, 635–644. [CrossRef]
19. Tang, D.; Wei, F.; Yang, N.; Zhou, M.; Liu, T.; Qin, B. Learning sentiment-specific word embedding for twitter sentiment classification. In Proceedings of the 52nd ACL Conference, Baltimore, MD, USA, 23–25 June 2014; Volume 1, pp. 1555–1565.
20. Cagliero, L.; Garza, P. Improving classification models with taxonomy information. *Data Knowl. Eng.* **2013**, *86*, 85–101. [CrossRef]

21. Škrlj, B.; Kralj, J.; Lavrač, N. CBSSD: Community-based semantic subgroup discovery. *J. Intell. Inf. Syst.* **2019**, 1–40. [CrossRef]
22. Xu, N.; Wang, J.; Qi, G.; Huang, T.S.; Lin, W. Ontological random forests for image classification. In *Computer Vision: Concepts, Methodologies, Tools, and Applications*; IGI Global: Hershey, PA, USA, 2018; pp. 784–799.
23. Elhadad, M.K.; Badran, K.M.; Salama, G.I. A novel approach for ontology-based feature vector generation for web text document classification. *Int. J. Softw. Innov.* **2018**, *6*, 1–10. [CrossRef]
24. Kaur, R.; Kumar, M. Domain ontology graph approach using Markov clustering algorithm for text classification. In Proceedings of the International Conference on Intelligent Computing and Applications, Madurai, India, 14–15 June 2018; Springer: Berlin, Germany, 2018; pp. 515–531.
25. Ristoski, P.; Faralli, S.; Ponzetto, S.P.; Paulheim, H. Large-scale taxonomy induction using entity and word embeddings. In Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, 23–26 August 2017; pp. 81–87.
26. Liu, Q.; Jiang, H.; Wei, S.; Ling, Z.H.; Hu, Y. Learning semantic word embeddings based on ordinal knowledge constraints. In Proceedings of the 53rd ACL Conference and the 7th IJCNLP Conference, Beijing, China, 26–31 July 2015; Volume 1, pp. 1501–1511.
27. Bian, J.; Gao, B.; Liu, T.Y. Knowledge-powered deep learning for word embedding. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Nancy, France, 15–19 September 2014; Springer: Berlin, Germany, 2014; pp. 132–148.
28. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*; Curran Associates, Inc.: Red Hook, NY, USA, 2015; pp. 649–657.
29. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
30. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.
31. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
32. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*; Curran Associates, Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
33. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1,
34. Gal, Y.; Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*; Curran Associates, Inc.: Red Hook, NY, USA, 2016; pp. 1019–1027.
35. Cheng, J.; Dong, L.; Lapata, M. Long short-term memory-networks for machine reading. *arXiv* **2016**, arXiv:1601.06733.
36. Graves, A.; Mohamed, A.R.; Hinton, G. Speech recognition with deep recurrent neural networks. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 26–31 May 2013; pp. 6645–6649.
37. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Gated feedback recurrent neural networks. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2067–2075.
38. Kowsari, K.; Heidarysafa, M.; Brown, D.E.; Meimandi, K.J.; Barnes, L.E. Rmdl: Random multimodel deep learning for classification. In Proceedings of the 2nd International Conference on Information System and Data Mining, Lakeland, FL, USA, 9–11 April 2018; pp. 19–28.
39. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
40. Kowsari, K.; Brown, D.E.; Heidarysafa, M.; Meimandi, K.J.; Gerber, M.S.; Barnes, L.E. Hdltex: Hierarchical deep learning for text classification. In Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 364–371.
41. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
42. Miller, G.A. WordNet: A lexical database for English. *Commun. ACM* **1995**, *38*, 39–41. [CrossRef]

43. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* **2015**, arXiv:1511.07289.
44. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
45. Chollet, F. Keras. 2015. Available online: https://github.com/fchollet/keras (accessed on 20 March 2019).
46. Robnik-Šikonja, M.; Kononenko, I. Theoretical and empirical analysis of ReliefF and RReliefF. *Mach. Learn.* **2003**, *53*, 23–69. [CrossRef]
47. Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 27. [CrossRef]
48. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
49. 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.
50. Walt, S.V.D.; Colbert, S.C.; Varoquaux, G. The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.* **2011**, *13*, 22–30. [CrossRef]
51. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
52. Benavoli, A.; Corani, G.; Demšar, J.; Zaffalon, M. Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *J. Mach. Learn. Res.* **2017**, *18*, 2653–2688.
53. Hong, J.; Fang, M. *Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts*; Technical Report; Stanford University: Stanford, CA, USA, 2015.
54. Zhang, H.; Xiao, L.; Chen, W.; Wang, Y.; Jin, Y. Multi-task label embedding for text classification. *arXiv* **2017**, arXiv:1710.07210.