Matching, Re-ranking and Scoring: Learning Textual Similarity by Incorporating Dependency Graph Alignment and Coverage Features

Sarah Kohail and Chris Biemann

Language Technology Group
Computer Science Department
Universität Hamburg
Hamburg, Germany
{kohail, biemann}@informatik.uni-hamburg.de

Abstract. In this work, we introduce a supervised model for learning textual similarity, which can identify and score similarity between a set of candidate texts and a given query text. By combining dependency graph similarity and coverage features with lexical similarity measures using neural networks, we show that most relevant documents to a given text can be more accurately ranked and scored than if the lexical similarity measures were used in isolation. Additionally, we introduce an approximate dependency subgraph alignment approach allowing node gaps and mismatch, where a certain word in one dependency graph cannot be mapped to any word in the other graph. We apply our model to two different applications, namely re-ranking for improving document retrieval precision on a new dataset, and automatic short answer scoring on a standard dataset. Experimental results indicate that our approach is easily adaptable to different tasks and languages, and works well for long texts as well as short texts.

1 Introduction

Semantic Textual Similarity (STS) measures the degree of semantic equivalence between a given pair of text. It also determines the notion that some texts are more similar than others. Measuring textual similarity, resulting from paraphrasing or summarization, may improve language understanding for many Natural Language Processing (NLP) applications, ranging from Information Retrieval (IR) [1] and machine reading comprehension [2, 3] to question answering [4] and short answer scoring [5, 6].
In this paper, we address the problem of assessing STS of alternative versions of candidate texts that have a varying degrees of similarity to a given query text. Specifically, we try to examine the impact of using dependency graph similarity and coverage features, and leverage supervised machine learning techniques in order to improve the relevancy identification and scoring. We also present an approximate subgraph alignment approach to find a subgraph in the candidate text dependency graph that is similar to a given query text dependency graph, allowing for node gaps and mismatches, where a certain word in one dependency graph cannot be mapped to any word in the query text graph, as well as graph structural differences. We evaluate our method on two tasks:

re-ranking for information retrieval and automatic short answer grading.
The remainder of this paper is organized as follows. In Section 2, we briefly discuss related work. Section 3 describes our textual similarity model and feature representation. In section 4, we show the performance of our method and analyze results. Section 5 concludes and discusses further future work.

2 Related Work

Researchers have made substantial progress on STS motivated by the annual SemEval competitions [7–9]. Most of state of the art approaches often focus on training regression models on traditional lexical surface overlap features. Recently, deep learning models have achieved very promising results; the top three performing systems from SemEval STS 2016 used deep learning based models [10–12]. However, STS remains a hard problem when it comes to texts, which have both variable length and complex dependency structure [13].
To ensure that our method is generalizable over different languages and various text lengths, we evaluate our method using two different tasks, namely re-ranking for improving document retrieval, and automatic short answer grading.
Examining the effect of results ranking, Jansen and Spink [14] observed that most users do not browse results beyond the first page and the higher the document placement in the first page results, the more likely a user is to read that document. By minimizing the huge amount of relevant results to few highly relevant to the users' query, and re-ranking them to appear in the upper top rank, users are more likely to find their goal quickly and easily [15].
Since short texts might not contain sufficient statical information or syntax patterns, multiple evaluations have been proposed to operate for short texts separately [16–18], while Pilehvar and Navigli [19] investigate a unified approach to semantic similarity that operates at multiple levels.
The task of automatic short answer grading is to assess short natural language answers based on their similarity with expert-provided correct answers. Mohler et al. [6] train support vector machine (SVM) on a combination of graph-based alignment and lexical similarity measures to score short students answers using a 5-point scale. They find that the supervised model in this task outperforms the unsupervised model [5].
Numerous approaches have used the dataset of Mohler et al. [6] as a benchmark to evaluate their methods. We mention two recent comparable works, [20] and [21], which we use later in our comparative study. Ramachandran et al. [20] adopt a mechanism to automate the generation of regexp text patterns from the reference expert answer as well as top-scoring student answers, to capture the structural and semantic variations of good answers. Sultan et al. [21] train supervised model, namely a ridge regression model, on a set of similarity and word embeddings features for the task of short answer grading. They apply question demoting (QD) technique in an attempt to reduce the advantage of repeating words provided in the question by re-computing similarity features after removing these words from both the reference answer and the student response. Their ablation study shows that applying question demoting results in 0.021 correlation improvement and 0.016 reduction in Root Mean Square Error (RMSE).

3 Learning Textual Similarity

Given query text q and candidate text d, textual similarity captures the fact of how much a candidate text d conveys the same information as a query text q.
In this research, we employ two sets of features, similarity features and coverage features. The novel contribution of this work is constituted by three feature types: dependency structure features, expansion features and coverage features. In the following subsection, we describe each of these features in more details.

3.1 Similarity Features

Similarity is measured by the shared feature types between two texts a query text q and a candidate text d. For both texts d and q, we create a vector representation $\boldsymbol{d}$ and $\boldsymbol{q}$ for various feature types. Each entry in one vector corresponds to the existence/presence (i.e $d_i/q_i \in \{0,1\}$), frequency (i.e $d_i/q_i \in \mathbb{N}$) or Tf-Idf measure (i.e $d_i/q_i \in \mathbb{R}$) of a given feature type in a text. After removing stopwords, we consider the following feature types:

Bag of Words (BOW): We represent the content of each text by a bag of words. In this case, similarity is measured by the shared vocabulary between both d and q. We also employ a second version of this feature using stemmed words.

Topic Distribution: We also model each document as a vector of topics using Latent Dirichlet Allocation model (LDA) [22].

Dependency Structure: Another important similarity measure is dependency parse structure similarity. Based on the work by [23], we aggregate individual dependency relations obtained from a parser, weigh them with Tf-Idf and produce a graph which contains the highest-ranked content words and their dependency relations. For a text d, dependency graph $G_d = \{V_d, E_d\}$, where $V_d = \{w_1, ..., w_N\}$ represents the content words in a text, and E_d is an set of edges, where each edge e_{jk} represents a directed dependency relation between w_j and w_k. Written also as a list of triples as follows: "w_j" $\rightarrow$ "w_k"$[label =$ "e_{jk}"$]$.
A generated dependency graph is then filtered according to the following three conditions:
- Tf-Idf (w_j,d) $\geq \alpha$ or Tf-Idf (w_k,d) $\geq \alpha$
- Tf-Idf (w_j w_k,d) $\geq \beta$
- Tf-Idf (e_{jk} w_j w_k,d) $\geq \lambda$

where α, β and λ are ≥ 0. When α, β and $\lambda = 0$, no Tf-Idf filtering is applied. For q, dependency graphs are generated, and filtered in the same manner.
Similarity is then measured on these three levels by representing each text as a vector of words, pairs and relations.

Named Entities: In this case, we measure similarity based on named entities terms only.

Expansion Features: Since the variability of language allows expressing the same concepts, entities and facts in different words, measuring similarity purely based on exact word matching does not fully capture conceptual matching. We expand content words, i.e. (common and proper) nouns, adjectives, verbs and adverbs in each text text using the Distributional Thesaurus (DTs) from [24].

Once these vectors are constructed the similarity between each d and q texts pair can be measured using cosine similarity using the following equation:

$$cos(\boldsymbol{q}, \boldsymbol{d}) = \frac{\sum_{i=1}^{N} q_i d_i}{\sqrt{\sum_{i=1}^{N} {q_i}^2}\sqrt{\sum_{i=1}^{N} {d_i}^2}} \qquad (1)$$

where N is the dimension of q and d. Vectors are (length-) normalized by dividing by the L_2 norm $\sqrt{\sum_{i=1}^{N} {q_i}^2}$ and $\sqrt{\sum_{i=1}^{N} {d_i}^2}$ of q and d respectively.

3.2 Coverage Features

As a text gets longer, term frequency factors increase, and thus having a high similarity score is likelier for longer than for shorter texts. IR research has shown that document length normalization is important to guarantee that documents are retrieved with similar chances as their likelihood of relevance regardless of their length [25]. The same applies to answers grading: longer answers should not receive unjustly higher scores. Normalizing vectors using the cosine L_2 norm has proven to have several limitations due to the use of the individual terms weights for text length normalization [26, 27]. This dependency is undesirable when the text includes infrequent terms with high Idf value, which can significantly increase the overall cosine normalization L_2 factor and cause inaccurate weighting for the other terms in the text. Accordingly, the new weights may not reflect the actual importance of the terms in content representation of the text. We try to solve this problem by incorporating a set of coverage features to measure the coverage of the query in the document.
Let $G_d = \{V_d, E_d\}$ and $G_q = \{V_q, E_q\}$ be the dependency graphs of d and q respectively. We measure coverage using the following equations:

Vocabulary Coverage: We calculate vocabulary coverage by computing the number of one-to-one nodes correspondence between both q and d dependency graphs divided by the overall number of nodes in the query text q dependency graph, as in the following equation:

$$\frac{|V_d \cap V_q|}{|V_q|} \qquad (2)$$

Relation Coverage: We calculate relation coverage by computing the number of one-to-one edges (triple) correspondence between both q and d dependency graphs divided by the overall number of edges in the query text q dependency graph:

$$\frac{|E_d \cap E_q|}{|E_q|} \qquad (3)$$

Pair Coverage: As in relation coverage, however in this case, we ignore the relation type and edge direction.

Graph Coverage: Before we present more details about how graph coverage features are measured, however, we first need to introduce our approximate dependency sub-graph alignment methodology.

The idea is to find a subgraph $G_s = \{V_s, E_s\}$, where $G_s \subseteq G_d$, that is approximately similar to a query text graph G_q. Algorithm 1 shows the pseudo-code of dependency sub-graph approximate matching algorithm.

Input: G_d, G_q, Threshold t
Output: G_s
$V_{intersection} \leftarrow \{V_d \cap V_q\}$;
for $j \leftarrow 1$ ***to*** $|V_{intersection}| - 1$ **do**
  for $k \leftarrow j + 1$ ***to*** $|V_{intersection}|$ **do**
    $Path \leftarrow dijkstra.getPath(G_d, w_j, w_k)$;
    if $Path \neq null$ ***and*** $Path.size(w_j, w_k) \leq t$ **then**
      $G_s \leftarrow G_s \cup Path$;
    end
  end
end

Algorithm 1: Dependency sub-graph approximate alignment methodology.

First, we obtain the nodes intersection $V_{intersection}$ between both q and d dependency graphs. We then find the shortest path between every pair (w_j, w_k) of vertices belongs to the $V_{intersection}$ set in the candidate text dependency graph using Dijkstra's algorithm [28]. Each edge was given a weight of 1 and edges directions are ignored during the process of the algorithm. Due to linguistic variation, we may not find a sub-graph that match the exact query text graph, however, we might find a sub-graph that match the query text graph approximately. We define a threshold parameter t to allow node gaps and mismatch in the case where some nodes in the query text cannot be mapped to any nodes in the candidate text graph. If the shortest path size (i.e number of edges between w_j and w_k) is less than or equal t, the path will be added to the sub-graph G_s. By setting t to a value greater than 1, it is much more likely to capture syntactic variations. Figure 1 shows examples of sub-graph matching from the dataset of [6].
The resulting G_s is used to measure two graph coverage features as follows:

$$\frac{|E_s|}{|E_d|} \qquad (4)$$

$$\frac{|E_s|}{|E_q|} \qquad (5)$$

Since a much more relevant candidate text is much more likely to have a larger overlap with the query text than other less relevant candidates, this may well tend to improving relevant documents similarity assessment.

4 Applications and Analysis

We evaluate our method for two different tasks: re-ranking for article summary matching, and short answer grading.

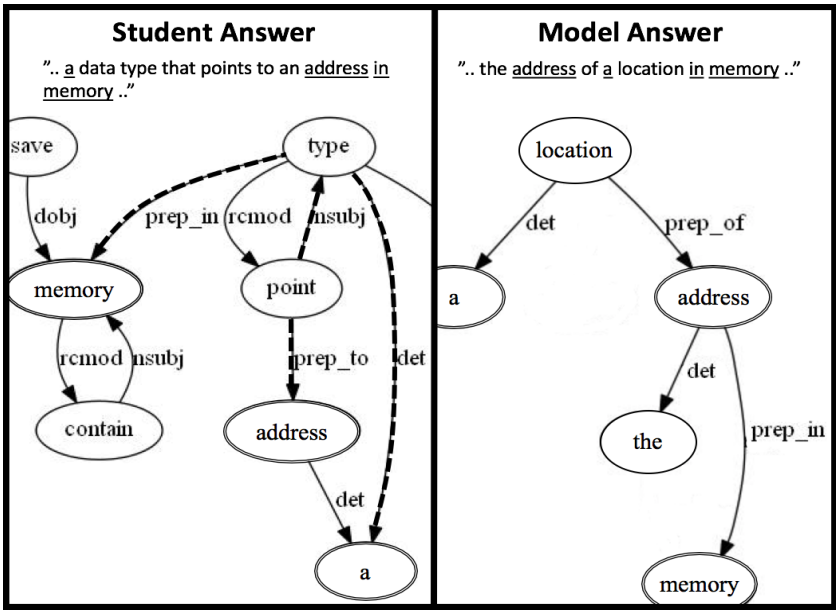

Fig. 1. Approximate sub-graph matching illustration. Example is taken from [6]. Given a model and a student candidate answer, double lined nodes represents the shared words between both answers and connections between words represents dependency relations. Direction and dependency types are ignored. Algorithm 1 uses the dependency structure similarity of local neighborhoods within Shortest Path (SP $\leq$ **t**), to find an approximate sub-graph that match the model answer. The selected subgraph is highlighted by bold dotted lines. In this example **t = 3**.

4.1 Re-ranking for Article-Summary Matching

Similar to [15], we utilize the ranking output of an IR system to do re-ranking. We choose to select the top n relevant documents ranked by Lucene[1] and incorporate our features to improve documents ranking precision.

Lucene Ranking Our re-ranking method is build on the top of Lucene. Lucene offers an open source information retrieval library, which provides IR-related tasks like indexing, querying, language analysis, results scoring and retrieval. Figure 2 shows how our re-ranking setup with Lucene works. Depending on the chosen language analyzer, the documents are internally tokenized, stemmed and filtered for stopwords. The analyzer preprocesses and extracts the terms on which the searching can be done. The terms are then indexed into a format that facilitates rapid searching. When a query is issued, it gets analyzed and relevant results are selected from the collection by matching the query against the index. Finally, relevant documents are scored according to the following equation:

$$Score(d,q) = \sum_{w \in q} tf(w,d) \times idf(w) \times norm_{length}(d) \qquad (6)$$

[1] `https://lucene.apache.org/`

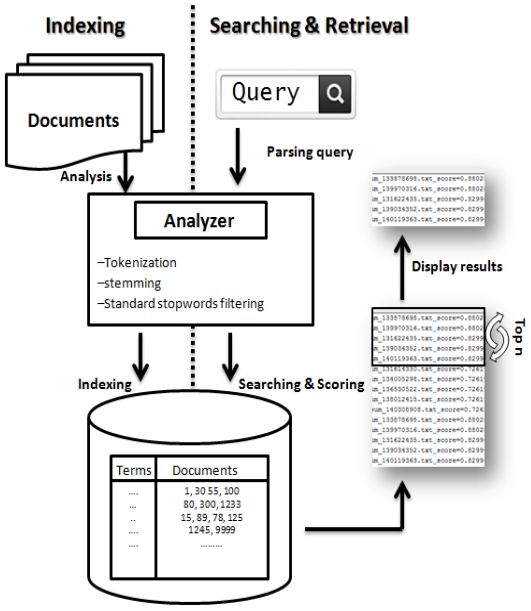

Fig. 2. Re-ranking top-n results of a retrieval system.

where $tf(w, d)$ is the word w frequency in document d, $idf(w)$ is the inverse number of documents in which word w appears, and $norm_{length}(d)$ is the length normalization factor for document d.

Dataset We used a set of 37,164 German news articles collected from Spiegel Online over 2015[2]. It includes German online news articles from different genres like sports, politic, economics, health, entertainment, etc. We remove images/video only articles, filter out irrelevant information like: source agency, date and translator name, clean HTML tags and extract only articles with summaries. The resulting corpus consists of 1130 (article, summary) pairs. Each of these articles has one corresponding summary that was created manually by the article author. Summaries are abstractive, which involves paraphrasing the facts from the original article using new novel sentences. Length ranges are [45-950] and [1185-9560] characters, and [7-107] and [216-1390] words, for summaries and articles respectively. The news are highly correlated due to the short one-year interval. Similar events are discussed in different contexts, therefore simple features like word frequency would not be able to discern the correct summary from summaries of articles on closely related topics. Thus, this dataset is suited for testing the capability of methods that assess a certain semantic understanding of texts – as opposed to e.g. the DUC datasets[3], where we found string-based matching to yield almost perfect scores in a preliminary experiment. The remaining articles, which have

[2] http://www.spiegel.de/
[3] http://duc.nist.gov/

no corresponding summaries, were used as a background corpus for Tf-Idf calculation and topic model training.
To create our dataset, we index the articles and use the summaries as a query to retrieve the articles. Then, we re-rank the n top-ranked documents returned by Lucene. We label the correct matching summary-article pairs as "1" and "0" otherwise. Overall, we have 5650 examples and 11300 examples, in the cases where $n = 5$ and $n = 10$ respectively. We apply the same process with the summaries indexed and the articles as queries. Table 1 shows Lucene retrieval results for both settings. Precision at n reports the fraction of documents ranked in the top n results that are labeled as relevant. Note that not all summary-article pairs could be correctly retrieved in all cases and the retrieval performance is not equally effective for retrieving texts and summaries, since summaries are much shorter and thus contain less distinctive words.

P@n	**Sum/Text**	**Text/Sum** [4]
P@1	1029 (0.9106)	887 (0.7849)
P@3	1111 (0.9831)	1021(0.9035)
P@5	1122 (0.9929)	1052 (0.9309)
P@10	1128 (0.9982)	1090 (0.9646)
P@20	1128 (0.9982)	1114 (0.9858)
P@50	1129 (0.9991)	1123 (0.9938)
P@100	1129 (0.9991)	1129 (0.9991)

Table 1. Lucene retrieval performance. Retrieval is evaluated by P@n. The number in parenthesis shows the overall P@n for the entire dataset of 1130 article-summary pairs. **Sum/Text:** summaries on index, articles for retrieval. **Text/Sum:** articles on index, text for retrieval

Experimental Setup For computing features, we use the implementation provided by [29] for topic modeling, and [30] for named entities extraction. Dependency graphs for both queries and documents are generated using the German collapsed parser by [31], and filtered using Tf-Idf thresholds in three levels. By manual inspection, α, β and λ are set to 10, 5 and 2 respectively. For lexical expansions features, we obtain the top 10 DT expansions using the JoBimText API[5].
Once the similarity and coverage scores have been computed for each summary-article pairs, we use a cost-sensitive Multilayer Perceptron (MLP) neural network to handle the imbalance between positive and negative examples. False positives are assigned a larger cost than false negatives, so the classifier would not be biased toward negative instances. The cost of false negatives is fixed to 1. We explore different costs to find the best cost using the validation set for false negatives class. We have found that a cost of $n - 1$ performs the best across all the training/validation rounds.
We run different experiments with different MLP structures and learning parameters. For evaluation, we use 5-fold cross-validation. We choose the model which provides

[5] `www.jobimtext.org/jobimviz-web-demo/api-and-demo-documentation/`

the best overall accuracy with a balanced classification error rate and stable performance scores between the two classes, which was determined on a smaller version of the dataset in preliminary experiments. The network structure includes 3 hidden layers,

	Sum/Text				Text/Sum			
Features	Precision	Recall	F-Measure	TP	Precision	Recall	F-Measure	TP
All	**0.916**	**0.894**	**0.899**	**1026**	**0.915**	**0.889**	**0.896**	**960**
BOW	0.890	0.853	0.864	1013	0.893	0.850	0.862	929
Dependency	0.887	0.838	0.850	1003	0.890	0.843	0.856	921
Coverage	0.893	0.853	0.864	919	0.850	0.861	0.862	917
Cov+Dep	0.904	0.874	0.882	1011	0.904	0.870	0.880	942
All/(Cov+Dep)	0.902	0.865	0.874	1023	0.901	0.860	0.870	948

Table 2. Binary relevancy classification results using MLP (n=5). Results shows binary relevancy classification precision (P@1), recall, F-measure and true positives.

	Sum/Text				Text/Sum			
Features	Precision	Recall	F-Measure	TP	Precision	Recall	F-Measure	TP
All	**0.951**	**0.929**	**0.936**	**1038**	**0.949**	**0.923**	**0.931**	**994**
BOW	0.938	0.894	0.908	1024	0.936	0.889	0.904	971
Dependency	0.933	0.879	0.895	1014	0.933	0.883	0.899	954
Coverage	0.930	0.881	0.896	972	0.935	0.888	0.903	962
Cov+Dep	0.945	0.918	0.926	1020	0.941	0.899	0.912	989
All/(Cov+Dep)	0.943	0.906	0.917	1035	0.942	0.903	0.915	989

Table 3. Binary relevancy classification results using MLP (n=10). Results shows binary relevancy classification precision (P@1), recall, F-measure and true positives.

with $(f + c)/2$ neurons in each layer, where f is the number of input features and c is the number of classes (i.e $c = 2$). Training time is set to 1000 epochs.
To re-rank, MLP[6] is configured to return a probability distribution of each class label. We re-rank the relevancy according to the descending ordering of the probability distribution of the positive class.

Results and Discussion The best report relevancy classification results are reported in Table 2 and Table 3. Since we are only aware of the correct article-summary pairs, we use P@1 only as a measure of performance. We also reported the recall, F-measure and true positives. We test the performance using different sets of features. From the results we observe the following: First, using all the features achieves the best performance over all measures in all cases. Second, using a combination of coverage and

[6] Learning rate = 0.5, momentum = 0.2

Features	n=5		n=10	
	Sum/Text	**Text/Sum**	**Sum/Text**	**Text/Sum**
All	1080 (0.962)	996 (0.946)	1077 (0.954)	1018 (0.933)
BOW	1064 (0.948)	987 (0.938)	1048 (0.929)	994 (0.911)
Dependency	1050 (0.935)	980 (0.931)	1037 (0.919)	976 (0.895)
Coverage	1038 (0.925)	962 (0.914)	1031 (0.914)	973 (0.892)
Cov+Dep	1079 (0.961)	991 (0.942)	1072 (0.950)	1010 (0.926)
All/(Cov+Dep)	1064 (0.948)	986 (0.937)	1048 (0.929)	993 (0.911)

Table 4. Re-ranking results using MLP probability distribution from different relevancy classification models when n=5 and n=10. The table shows the improvement in P@1 after the re-ranking. Numbers in brackets are the results of dividing by the number of cases, (1122,1052), (1128,1090), where the correct document is in the top 5 or top 10 Lucene results respectively, which form an upper bound.

dependency features lead to the second best performance and play a role in providing comparable performance to that obtained using all the features with an unnoticeable drop in true positives. Third, F-measures falls with average of 0.0215 (in four cases) when excluding these two features, and using each feature in isolation does not lead to any improvement, but achieves comparable results to the ones using BOW features. Forth, in most cases, we outperform Lucene P@1 in terms of true positives.
The improvement is most clearly seen when we use our neural networks models for re-ranking, see Tables 4.
In manual error analysis, we generally observe limitations on very short summaries that have no intersection with the text – most extremely noticed for an article on fashion history (866 words) with the (translated) summary "The suit is the uniform of gentlemen". Other errors could be addressed by a German compound splitter, as there are frequently compounds where only the parts match, such as "Ratenkreditangebote" and "Ratenkredite", other examples include derivational matches like "vorweihnachtliche" and "Vorweihnachtszeit", which could be addressed by an improved morphology component that also includes compound analysis.

4.2 Automatic Short Answers Grading

We provide a second evaluation for our method on automatic short answers grading.

Dataset We use the dataset by [6][7]. The dataset consists of 81 computer science questions on data structures course and 2273 student answers. The dataset was graded by two judges and normalized on a scale of 0..5 according to the extent to which the student answers are considered similar to the content of the correct answers. The reported inter-annotator agreement (IAA) between both judges is 0.586% (Pearsons ρ) and 0.659 Root Mean Square Error (RMSE).

[7] `http://web.eecs.umich.edu/~mihalcea/downloads.html\#saga`

Experimental Setup Dependency graphs for both questions and answers are based on collapsed dependencies from the Stanford Parser[8]. By manual inspection, we set α, β and λ to 4, 2 and 1 respectively. As average text length in this case is shorter, we choose smaller values.
We used New York Times articles within the years 1998-2000 as a background corpus for Tf-Idf calculation, and our topic model was trained using set of 36 million sentences from the recent English Wikipedia dump.
We train a MLP with one hidden layer using default parameters[9] for 5000 training epochs to increase stability. Following [6], we apply a 12-fold cross validation over the entire dataset for evaluation.

Results and Discussion Table 5 shows our results in comparison to previous approaches.

Features	ρ	**RMSE**
Tf-Idf	0.327	1.022
Lesk	0.462	1.050
Mohler et al. [6]	0.518	0.978
Inter-annotator Agreement (IAA)	0.586	0.659
Sultan et al. [21]	0.592	0.887
Sultan et al. [21] w/ Question Demoting	0.571	0.903
Ramachandran et al. [20]*	0.610	0.860
Our Method	**0.590**	**0.847**

Table 5. Comparing performance of different models trained on [6]. Comparison is based on Pearson's ρ correlation and Root Mean Square Error (RMSE). * results on a smaller test dataset, not directly comparable.

Our approach exhibits superior performance over existing models when evaluating on RMSE except for IAA, and we perform quite well in comparison to IAA and [21] in terms of Pearson's correlation. Although [20] report better results; however, their evaluation is based on much smaller test data (453 examples) and they use in-domain model training. As can be seen as well, our coverage and alignment features proven to has a great effect on improving the performance than when only considering BOW or Tf-Idf features in isolation.
Further manual error analysis shows that a substantial portion of the errors are due to unstructured answers and misspelling. Again, a more lenient matching mechanism, e.g. using edit distance or automatic spelling correction, might alleviate these errors.

[8] `http://nlp.stanford.edu/software/lex-parser.shtml`
[9] Learning rate = 0.3, momentum = 0.2

5 Conclusion

In this paper, we introduced a supervised approach for learning to rank and score a set of candidate texts that have varying degrees of similarity to a given query text. We showed that incorporating additional structural and content similarity features, coverage measures and lexical similarity from distributional thesaurus can produce better results than if each were used individually.

To enable interpretable similarity, we also developed an approximate dependency subgraph alignment algorithm. The idea is to find a subgraph in the candidate text dependency graph that is similar to a given query text dependency graph, allowing for syntactic variations.

To ensure that our method is generalizable over different languages and various text lengths, we evaluate our method using two different tasks, namely re-ranking for improving document retrieval, and short answer grading. Results indicate that our approach provide better or comparable performance to baseline and recent approaches.

In the future, we would like to improve the quality of our alignment algorithm by incorporating semantic similarity, which will help capturing synonyms and paraphrases. Further improvement would be to explore more lenient matching mechanisms to capture morphological variants and misspellings.

References

1. Amiri, H., Resnik, P., Boyd-Graber, J., Daumé III, H.: Learning text pair similarity with context-sensitive autoencoders. In: ACL, Berlin, Germany (2016) 1882–1892
2. Chen, D., Bolton, J., Manning, C.D.: A thorough examination of the cnn/daily mail reading comprehension task. In: ACL, Berlin, Germany (2016) 2358–2367
3. Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., eds.: Advances in Neural Information Processing Systems 28, Montreal, Canada (2015) 1693–1701
4. Weston, J., Bordes, A., Chopra, S., Mikolov, T.: Towards ai-complete question answering: A set of prerequisite toy tasks. CoRR (2015)
5. Mohler, M., Mihalcea, R.: Text-to-text semantic similarity for automatic short answer grading. In: EACL 2009, Athens, Greece (2009) 567–575
6. Mohler, M., Bunescu, R., Mihalcea, R.: Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In: ACL-HLT. HLT '11, Portland, Oregon, USA (2011) 752–762
7. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: SemEval, Montreal, Canada (2012) 385–393
8. Agirre, E., Banea, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Mihalcea, R., Wiebe, J.: Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In: SemEval, San Diego, California (2016) 497–511
9. Agirre, E., Gonzalez-Agirre, A., Lopez-Gazpio, I., Maritxalar, M., Rigau, G., Uria, L.: Semeval-2016 task 2: Interpretable semantic textual similarity, San Diego, California (2016) 512–524
10. Rychalska, B., Pakulska, K., Chodorowska, K., Walczak, W., Andruszkiewicz, P.: Samsung poland nlp team at semeval-2016 task 1: Necessity for diversity; combining recursive autoencoders, wordnet and ensemble methods to measure semantic similarity. In: SemEval, San Diego, California (2016) 602–608

11. Brychcín, T., Svoboda, L.: Uwb at semeval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information. In: SemEval, San Diego, California (2016) 588–594
12. Afzal, N., Wang, Y., Liu, H.: Mayonlp at semeval-2016 task 1: Semantic textual similarity based on lexical semantic net and deep learning semantic model. In: SemEval, San Diego, California (2016) 674–679
13. Mueller, J., Thyagarajan, A.: Siamese recurrent architectures for learning sentence similarity. In: Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona USA (2016)
14. Jansen, B.J., Spink, A.H.: Investigating customer click through behaviour with integrated sponsored and nonsponsored results. International Journal of Internet Marketing and Advertising **5** (2009) 74–94
15. Hagen, M., Völske, M., Göring, S., Stein, B.: Axiomatic result re-ranking. In: CIKM 2016, Indianapolis, Indiana (2016)
16. Yang, S., Lu, W., Yang, D., Yao, L., Wei, B.: Short text understanding by leveraging knowledge into topic model. In: NAACL: HLT, Denver, Colorado, USA, Association for Computational Linguistics (2015) 1232–1237
17. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '15, Santiago, Chile (2015) 373–382
18. Gu, Y., Yang, Z., Zhou, J., Qu, W., Wei, J., Shi, X.: A fast approach for semantic similar short texts retrieval. In: ACL, Berlin, Germany (2016) 89–94
19. Pilehvar, M.T., Navigli, R.: From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. Artificial Intelligence **228** (2015) 95–128
20. Ramachandran, L., Cheng, J., Foltz, P.: Identifying patterns for short answer scoring using graph-based lexico-semantic text matching. In: Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications, Denver, Colorado, USA (2015) 97–106
21. Sultan, M.A., Salazar, C., Sumner, T.: Fast and easy short answer grading with high accuracy. In: NAACL: HLT, San Diego, California (2016) 1070–1075
22. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research **3** (2003) 993–1022
23. Kohail, S.: Unsupervised topic-specific domain dependency graphs for aspect identification in sentiment analysis. In: Student Research Workshop Associated with RANLP 2015, Hissar, Bulgaria (2015) 16–23
24. Biemann, C., Riedl, M.: Text: Now in 2D! a framework for lexical expansion with contextual similarity. Journal of Language Modelling **1** (2013) 55–95
25. Albalate, A., Minker, W.: Semi-Supervised and Unervised Machine Learning: Novel Strategies. John Wiley & Sons (2013)
26. Buckley, C., Singhal, A., Mitra, M., Salton, G.: New retrieval approaches using smart: Trec 4. In: TREC, Gaithersburg, Maryland (1995) 25–48
27. Singhal, A., Salton, G., Buckley, C.: Length normalization in degraded text collections. In: Proceedings of Fifth Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada, USA (1996) 15–17
28. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269–271
29. Phan, X.H., Nguyen, C.T.: GibbsLDA++: A C/C++ implementation of latent Dirichlet allocation (LDA). (2007) `http://gibbslda.sourceforge.net`.
30. Benikova, D., Yimam, S.M., Santhanam, P., Biemann, C.: GermaNER: Free Open German Named Entity Recognition Tool. In: GSCL, Duisburg-Essen, Germany (2015) 31–38
31. Ruppert, E., Klesy, J., Riedl, M., Biemann, C.: Rule-based Dependency Parse Collapsing and Propagation for German and English. In: GSCL, Duisburg-Essen, Germany (2015) 58–66