Combining Lexical and Semantic Similarity Methods for News Article Matching

Mehmet Umut Sen
Donanim Haber, Sabanci University

Hakki Yagiz Erdinc
Donanim Haber, Dogus University

Burak Yavuzalp
Donanim Haber, Istanbul Technical University

Murat Can Ganiz
Verius, Marmara University

***Abstract*—Matching news articles from multiple portals with different narratives is a necessary step towards extensive modeling of online news flow. Although deep neural network based architectures are successfully applied on various semantic matching problems, their performance on matching long text documents with limited number of labeled training data points are insufficient. In this work, we argue that combining different lexical matching scores together with word embedding similarity scores result in successful news matching even with simple thresholding. We also train classifiers on labeled data using scores as features along with length and time features and improve the results further. Feeding the classifiers with additional features obtained by matching title and spot fields of news articles paves the way for a more robust model.**

***Index Terms*—news matching, semantic matching, semantic similarity, lexical matching, word embeddings, jaccard similarity distance**

I. Introduction

As the amount of news portals had grown and the importance of publishing rate had increased due to the rapid sharing of news on social media, a necessity for organizing and summarizing vast amounts of news items has emerged. Modelling news events using multiple sources would be useful for creating summarizing stories of long term event sequences as well as detecting false news. A necessary step of this news modelling, that we focus on in this paper, is matching news articles from different portal sources that correspond to the same event.

News matching can be considered as a sub-problem of semantic similarity (SS) which aims to model the similarity of different textual elements regarding meaning of those elements [1], [2]. Semantic similarity is mostly studied for the information retrieval problems such as question answering, text summarization and search. These problems, although numerous models have been proposed and applied to them [2]–[4], are fundamentally different than the news matching problem in two ways: First, at least one of the two items in the pair are short text documents, such as questions, queries, summaries, etc. However, news articles usually are long text documents. Second, unlike in semantic matching problems, queries are commutative in the news matching problem, i.e. both items in the query pair are news articles.

Despite various architectures and models on different semantic similarity problems, matching long text documents is still a challenging problem [1]. In this paper we investigate the performance of simpler lexical matching based scores for the news matching problem. We argue that, even though two matched news articles have different narratives, the number of keywords and entities that define the event of concern is reasonably high for long articles and that is in favor for the lexical matching based scoring methods.

In this work; we first experiment on various lexical matching methods, which are unsupervised methods except for the threshold parameter, and show improved performance over previous work on semantic similarity which leverage word embeddings. In addition, combining lexical matching scores and cosine similarity scores of different word embedding methods, namely Word2Vec [5] and FastText [6]–[8], improves the performance further.

We also experiment with supervised classifiers that are trained with obtained similarity scores as features along with other features such as the time difference and length difference of news articles, and obtain improved performance.

As mentioned before, some matched news articles may have different level of details about an event. This imbalance creates noise for the lexical matching scores. To alleviate this problem, we obtain additional scores between other fields of the two news articles, namely title and spot. We show improved performance using these additional field scores.

In Section-II we summarize some previous work. In Section-III we define the problem, explain the unsupervised and supervised methods that we applied. In Section-IV we first describe the dataset that we collected, define the experimental setup, give results and discuss them. We give concluding remarks and future work at Section-VI. [1]

II. Related Work

As discussed at Section-I, most existing works on semantic similarity focus on problems such as question answering, text summarization and search [2]. Traditional methods exploit lexical databases such as WordNet [9], or any other structured

[1]The dataset and codes are availeble at: https://github.com/donanimhaber/newsmatching

semantic knowledge sources as for the biomedical text matching problem [10], [11], to measure the semantic similarity of words. However, using such external lexical resources is not practically applicable to our problem, because technical terms and named entities in the news items are of vast domain and evolve in time. In addition, high quality WordNet databases are not available in all languages.

More recent methods of semantic similarity use word embeddings that are obtained by unsupervised training on large corpora. Kenter et al. apply BM25 algorithm [12] to word embbeddings, along with other meta-features, for semantic similarity of short texts [13]. We compare with this algorithm in our experiments. Kusner et al. introduce Word Mover's Distance (MVD) which casts the dissimilarity of two sets of word embeddings as Earth Mover's Distance transportation problem, where each word embedding from one document moves to embeddings in the other document with some proportions and the minimum distance is considered as the dissimilarity [14]. They report that removing one constraint in the optimization problem results in slightly lower performance but the computation time is highly reduced. We compare with this similarity algorithm, which they call Relaxed WMD (RWMD), in our experiments.

Recent supervised models mostly employ Deep Neural Network (DNN) architectures to achieve better accuracy on these tasks than the traditional methods [3], [4], [15]. Pang et al. obtains a matrix with word embeddings that contains interactions of word pairs among the two documents and treat the matrix as an image to feed into a Convolutional Neural Network (CNN) with dynamic pooling layer [3]. Hu et al. uses a similar architecture with max pooling and apply their model to various Natural Language Processing (NLP) problems such as sentence completion, matching a response to a tweet, etc. [4]. The only work, to the best of our knowledge, that deals with matching long texts of events is the work of Liu et al. [1]. For the purpose of efficient learning with Neural Networks for long documents, they first embed these long documents into a *concept* graph, where nodes in the graph represent different concepts. Each sentence in the document is assigned to a different concept. Then, graph pairs are fed into a *Siamese Encoded Graph CNN*. They obtain better results for the news matching problem than the general similarity matching models.

These DNN models perform poorly in our problem because they require large amount of labeled data, whereas in our dataset a small ($\approx 2K$) number of news articles are labeled. Our trials with smaller DNN models performed poorly on our dataset, therefore we discard these results from the Experiments Section.

III. Method

A. Problem Definition

We tackle the problem of matching news with the same or very similar content of different portal sources. Document matches are postulated as news stories depicting the same event. We formulate the problem as inputting a pair of documents to the model and outputting a binary decision as "same" or "different".

Each document has the following fields:

1) Title: Heading of the news story.
2) Spot: Sub-heading of the story.
3) Body: Details and the main content of the story.
4) Date-time: Last modified date and time of the story.

We assume that some fields can be empty for some documents. We created another field named "text" that is the concatenation of "title", "spot" and "body" fields and used this field instead of "body". So "text" field is all the textual content of the document, and we did not use the "body" since it is usually very close to "text" because of the short lengths of "title" and "spot".

Textual fields (title_i, spot_i and text_i) of a document doc_i are sequences of words, for example $S^{(i)}_{\text{title}} = [w_0, w_1, \ldots, w_N]$.

First we describe our text based methodologies for unsupervised scoring of document pairs. In the following section, we describe the supervised setting and methods.

B. Unsupervised Scoring

Given two documents, doc_1 and doc_2, we calculate four different similarity scores for all three fields "title", "spot" and "text". Three of these scores are based on lexical matching. Another scoring uses word embeddings to capture semantic similarity.

1) Lexical Matching Scores: Jaccard Similarity Coefficient (JSC) [16] with unique words in the field is obtained as follows:

$$\text{JU}(\text{field}_i, \text{field}_j) = \frac{\left|U^{(i)}_{\text{field}} \cap U^{(j)}_{\text{field}}\right|}{\left|U^{(i)}_{\text{field}} \cup U^{(j)}_{\text{field}}\right|}, \tag{1}$$

where $U^{(i)}_{\text{field}}$ is the set of unique words in the sequence $S^{(i)}_{\text{field}}$, $|A|$ is the cardinality of a set A, $\cap$ is the intersection operator and $\cup$ is the union operator.

We calculate JSC also with all words as opposed to unique words:

$$\text{JC}(\text{field}_i, \text{field}_j) = \frac{\left|C^{(i)}_{\text{field}} \cap C^{(j)}_{\text{field}}\right|}{\left|C^{(i)}_{\text{field}} \cup C^{(j)}_{\text{field}}\right|}, \tag{2}$$

where $C^{(j)}_{\text{field}}$ is obtained by enumerating words by the occurence count in the field. For example, for a field with words S_{field} = ["the", "cat", "sat", "on", "the", "mat"], the counted field set is S_{field} = {"the-1", "cat-1", "sat-1", "on-1", "the-2", "mat-1"} at which the second occurence of the word "the" is now different than the first occurence.

One issue we observed with these scorings is that for news pairs that give substantially different amount of details about an event results in poor JU and JC scores. Therefore we define another scoring function that replaces the denominator of the JU function with the length of the short document field:

$$\text{JS}(\text{field}_i, \text{field}_j) = \frac{\left|U^{(i)}_{\text{field}} \cap U^{(j)}_{\text{field}}\right|}{\min\left(\left|U^{(i)}_{\text{field}}\right|, \left|U^{(j)}_{\text{field}}\right|\right)}, \quad (3)$$

Even though we expect that JS would result in better scorings for truely matched pairs with a large difference in length, it may result in poor performance on other pairing cases, resulting in lower performance on overall dataset. But, we keep this scoring to be used as a feature with supervised classification models.

2) Word Embedding Scores: We use two different word embedding models: Word2Vec and FastText.

Word2Vec is a continuous word representation where each word is mapped to a low dimensional real vector [5]. We use the Skipgram model which trains with the objective of predicting the context words, i.e. nearby words, given an input word. A large unlabeled news articles corpus is used to train the model. After training, we obtain the vector of a given document by averaging vectors of the words contained in the document.

FastText is another continuous word representation method. Unlike previous unsupervised word embedding models such as Word2Vec [5], Doc2Vec [17], Glove [18]; FastText trains embeddings of character n-grams in addition to the word n-grams using skip-gram model [19]. It obtains the representation of a word by averaging over character n-gram vectors and the word vector. For example vector of the word "the" is obtained by averaging over the word vector "the" and the character n-gram vectors of "<t", "th", "he", "e>", "<th", "the", "he>" if only 2-grams and 3-grams are used.

Using character n-gram embeddings paves the way to dumping syntactic information into the representative vectors. This is particularly beneficial for morphologically rich languages such as Turkish, Finnish, Hungarian etc. In addition, it helps to deal with syntax errors and typos which occurs frequently in non-editorial texts.

Since the number of n-grams would be enormous for a reasonably large corpus and a reasonable selection of n in the n-gram such as 3 to 6; memory requirement of this algorithm would be very high. Mikolov et al. deal with this problem by randomly grouping n-grams using a hash function and using the same vector for n-grams that are in the same group [19].

FastText obtains the document vector by first calculating word vectors, normalizing them such that their l_2 norm is 1 and then averaging them.

We use the cosine similarity to obtain the word embedding score of a pair of document fields:

$$\text{WE}(\text{field}_i, \text{field}_j) = \frac{\mathbf{v}_i^T \mathbf{v}_j}{\|\mathbf{v}_i\| \, \|\mathbf{v}_j\|}, \quad (4)$$

where $\mathbf{v}_i$ is the FastText (FT) or Word2Vec (WV) vector of field_i and $\|.\|$ is the l_2 norm.

3) Thresholding: We obtain an optimal threshold for each scoring function and for each field using a labeled dataset of news articles. We use the threshold for which precision is equal to recall since the number of negative examples, i.e. pairs that do not match, is much higher than the number of positive examples. This corresponds to the threshold for which the number of *false negatives* (falsely classified as negatives) is equal to *false positives* (falsely classified as positives).

4) Combination with Weighted Averaging: We combine different scores by weighted averaging. First, we normalize each score using the corresponding threshold and the standard deviation:

$$\text{JU}_{\text{norm}}(\text{field}_i, \text{field}_j) = \frac{\text{JU}(\text{field}_i, \text{field}_j) - \text{thr}_{\text{JU-field}}}{\text{std}(\{\text{JU}(\text{field}_i, \text{field}_j)\}_{(i,j)\in X})} \quad (5)$$

where X contains the index pairs in the training data, $\text{thr}_{\text{JU-field}}$ is the optimal threshold for the method JU and the field *field*. Normalized scores for other fields are computed similarly. After normalization of individual scores, we obtain the combined score as follows:

$$\text{COMB}(\text{field}_i, \text{field}_j) = \frac{1}{5} \sum_{m \in \{\text{JU,JS,JC,FT,WV}\}} \sum_{f \in \{\text{title,spot,text}\}} w_{\text{f}} s^{(i,j)}_{fm} \quad (6)$$

where w_f is the weight of field f independent of the scoring method, $s^{(i,j)}_{fm}$ is the normalized score of documents i and j with the method m for the field f, such as $\text{JU}_{\text{norm}}(\text{title}_i, \text{title}_j)$ for field *title* and method *JU*. We choose the weights manually proportional to the average lengths of the fields in the training data, i.e. $w_{\text{title}} < w_{\text{spot}} < w_{\text{text}}$ and weights sum to one.

Optimal threshold for the combined score is obtained similar to those of previous scoring methods. However, since the normalized scores have the optimal threshold of 0, the resulting optimal threshold for the combination is very close to 0.

5) Compared Semantic Similarity Methods: We compare with two similarity scoring methods which employ word embeddings. First one is the Word Mover's Distance (WMD) [14] which minimizes the following constrained optimization problem:

$$\min_{\mathbf{T} \geq 0} \sum_{i,j} T_{i,j} c(i,j) \quad (7)$$

subject to

$$\sum_j T_{ij} = d_i \quad \forall i \quad (8)$$

$$\sum_i T_{ij} = d'_j \quad \forall j \quad (9)$$

where, $c(i,j)$ is the Euclidean Distance between embeddings of the i^{th} and j^{th} unique words of the first and second document respectively, d_i and d'_j are the frequencies of the corresponding words in the document. After finding optimal $\mathbf{T}$, the distance between documents is computed by

$\sum_{i,j} T_{i,j} c(i,j)$. Since solving above problem is computationally difficult, Kusner et al. relaxed this optimization problem by removing one of the constraints, obtain two easy-to-optimize problems one for each constraint (Relaxed WMD). Maximum of the two distances is used as the final distance. They show that RWMD obtains similar classification results with the WMD, therefore we obtained results using RWMD.

Kenter et al. [13] uses the following BM25 similarity algorithm [12] for short texts of S_1 and S_2:

$$f(S_1, S_2) = \sum_{w \in S_1} \text{IDF}(w) \frac{\text{sem}(w, S_2)(k_1 + 1)}{\text{sem}(w, s) + k_1(1 - b + b\frac{|S_2|}{L})} \tag{10}$$

where, IDF(w) is the Inverse Document Frequency, L is the average document length, $|S_1| \geq |S_2|$, b and k_1 are meta-parameters and sem is defined as follows:

$$\text{sem}(w, S) = \max_{w' \in S} \frac{\mathbf{v}_w^T \mathbf{v}_{w'}}{||\mathbf{v}_w^T||\, ||\mathbf{x}_{w'}||} \tag{11}$$

where, $\mathbf{v}_w$ is the word embedding of the word w.

C. Supervised Classification

We use the scores described in previous section as features to be fed into a classifier. In addition, we extract *length* features and *time* features. For *length* features, we use mainly two inputs: l_1 and l_2 which represent the lengths (number of words) of fields of the document pair. We extract the following *length* features for each field ("title", "spot", "text"):

- Minimum of lengths: $\min(l_1, l_2)$
- Maximum of lengths: $\max(l_1, l_2)$
- Absolute value of difference of lengths: $|l_1 - l_2|$
- Absolute value of the difference divided by maximum of lengths $|l_1 - l_2| / \max(l_1, l_2)$
- Maximum of the lengths divided by the minimum of lengths: $\max(l_1, l_2) / \min(l_1, l_2)$

In addition to the *text length* features, we extract *time* features which are the difference of the last modified times of the two news articles. We extract two features corresponding to time difference in hours and in days. These features provide significant information to the classifier since news articles are published mostly on the same day with the event of subject.

We have 16 score features, 15 textual length features and 2 time features which amounts to a total of 33 features. These features are then fed to various classifiers including *Random Forest* (RF), *Support Vector Machines* (SVM) and *Multilayer Perceptron* (MLP). Majority Voter (MV) classifier combination results are also reported.

IV. Experiments

In subsequent sections we describe the dataset, preprocessing of the texts, parameters and details of the implementation and give results and discuss them.

A. Labeled News Dataset

We evaluate the methods on a news articles corpus in Turkish whose URLs are obtained manually by searching for the articles of the same events on different news portals. Then we crawled the web pages to obtain the fields "title", "spot", "body" and "datetime".

There are in total 20 portal domains in the dataset and 2049 news items in total. News articles span approximately 6 months, but majority of the articles span 1 month. We obtained 693 groups for which news in the same group correspond to the same event, i.e. positively labeled. Each group contains 2.75 documents on average. We obtained a total of 1858 positive examples and randomly obtained $15,000$ negative examples which amounts to a total of 16858 examples. Average numbers of words in the dataset are 6.94 ± 2.82, 25.72 ± 13.86 and 205.4 ± 223.72 for the fields *title*, *spot* and *body* respectively where second arguments are the standard deviations.

We also use an unlabeled news corpus in Turkish of size ≈ 4.7 million for training the Word2Vec and FastText models which does not contain news articles of the labeled dataset. We apply the same preprocessing steps on the unlabeled as with the labeled dataset.

B. Preprocessing

We apply the following preprocessing steps to all text fields:

- Escape *html* character references.
- Remove *html* tags.
- Lowerize.
- Sentence tokenize using NLTK toolkit [20].
- Use morphological analysis and disambiguation with Zemberek toolkit [21] and get lemmas.
- Remove stop-words.

C. Settings

We use a vector dimension of 100 for the word embeddings. For Word2Vec, gensim toolkit is used with skipgram model and negative sampling [22]. Context window length parameter is chosen as 5, minimum count for filtering out words is set to 5 and training performed for 5 epochs. There were no improvement on the loss value after a few epochs.

For the FastText model; minimum count for filtering out words is set to 5, context window length parameter is chosen as 5, bucket size is chosen to be $2,000,000$ and from 3-grams up to (including) 6-grams are used for character n-gram embeddings. Training performed for 100 epochs.

For combination of different fields and lexical matching methods as in (6), we used the following weights: $w_{\text{title}} = 0.1$, $w_{\text{spot}} = 0.3$, $w_{\text{text}} = 0.6$.

Scores for missing fields are set to the mean of the corresponding field's scores in the dataset.

We used Random Forest (RF), Multilayer Perceptron (MLP) and Support Vector Machines (SVM) for the supervised classification. Models are implemented using the *sklearn* toolkit [23]. For the RF, 100 trees are used with 2 as

the minimum samples per split, 2 as the minimum samples per leaf, and Gini impurity as the split criterion. For MLP, we used 2 layers with 500 nodes each, *adam* solver, batchsize of 100 and Relu activations. Training is stopped when the loss is not decreased for at least $1e-4$ in 10 epochs. For the SVM classifier, we used *RBF* kernel and applied grid search for the penalty parameter (C) and the RBF kernel parameter (γ). We searched in $\{1e-7, 1e-5, 1e-3, 1e-1, 1, 10\}$ and $\{1, 10, 100, 1000, 10000, 100000\}$ for γ and C respectively.

We normalized all the classifier features to the range $[0, 1]$. 10-fold Cross Validation (CV) are applied to test the performances. For the feature that divides maximum length by the minimum, we used the maximum of the feature along the dataset if one of the field is missing and used 1 if both fields are missing.

For the BM25 algorithm, we applied grid search for the meta-parameters b and k_1 in $\{0.8, 1, 1.2, 1.5, 2, 2.5, 5, 10, 20, 50\}$ and $\{0, 0.001, 0.01, 0.1, 0.2, 0.5, 0.75, 0.9, 1\}$ respectively. For the RWMD method, we calculated Inverse Document Frequencies from the large unlabeled news corpora.

V. Results

Results for lexical matching based and word embedding based similarity scoring along with compared algorithms are shown at Table-V. Here, *FT* stands for FastText, *WV* stands for Word2Vec, *COMB.* is the average of different scores as in (6).

We obtained scores for all fields *title*, *spot*, *text* along with the combination of scores for these three fields, which is depicted with *Weighted Av.* in the table, similar to using (6) except with only the related method. *Combined* results are also computed similarly.

TABLE I
F1 Results of Unsupervised Methods for Different Fields

Method	title	spot	text	Weighted Av.
RWMD-FT	0.8299	0.9386	0.9263	0.9645
RWMD-WV	0.8465	0.8762	0.9440	0.9758
BM25-FT	0.7438	0.8665	0.8881	0.9333
BM25-WV	0.7508	0.8741	0.8994	0.9413
Cosine FT	0.8407	0.9182	0.9273	0.9537
Cosine WV	0.8423	0.9225	0.9177	0.9403
JU	0.8328	**0.9535**	0.9639	0.9833
JS	0.8280	0.9476	0.9459	0.9839
JC	0.8339	0.9533	0.9709	0.9839
COMB. (FT,WM)	0.8466	0.9241	0.9225	0.9499
COMB. (JU,JS,JC)	0.8341	0.9532	**0.9817**	**0.9887**
COMB. (ALL)	**0.8617**	0.9532	0.9726	0.9833

We conclude from the results that in general lexical matching works better than word embedding based similarity methods, except for the *title* field for which FastText works better. This shows that as the length of the text decreases, importance of semantic information increases for text matching. Another useful observations is that FastText achieves higher performance than the Word2Vec model.

Compared algorithm RWMD outperforms cosine similarity of word embeddings, but performs worse than the lexical matching based results. BM25 algorithm does not work well even though extra IDF information is incorporated.

Even though *title* and *spot* fields result in lower scores than the *text* field, score level combination of three fields (*all* in the table) achieves higher performance than the *text* field itself, even though *text* field already contains the *title* and *spot* in its content. This is expected since some news pairs have a high difference in the amount of details and titles or spots may result in noise-free match.

Among different lexical matching methods (*JU*, *JS*, *JC*), *JU* performs the best although results are close to each other. Results for score level combination are depicted with *COMB.* at the table. Best performing method is the score level combination of lexical matching methods, i.e. *JU*, *JS* and *JC* with the weighted averaging of fields. However, word embedding based combination works better for the *title* field.

We show histograms of scores of negative and positive pairs for some methods and fields at Figure-V. Note that we use the right y-axis for positive class for clear visualization since the number of positive examples are much lower in the dataset. We observe that lexical matching based methods result in closer to uniform histograms than word embedding based methods. However, the combined method yields more Gaussian like distribution for the positive examples. We also see higher interference between the positive and negative classes at the *title* histogram than the *text* histogram of FT.

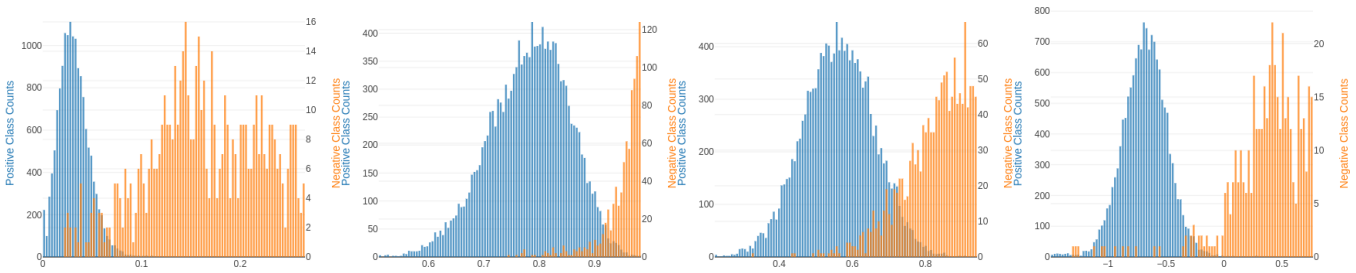
TABLE II
Results for Supervised Methods

Method	F1	Prec.	Recall	Acc.
WE-MLP	0.9895	0.9919	0.9871	0.9977
WE-RF	0.9879	0.9876	0.9882	0.9973
WE-SVM	0.9922	0.9935	0.9909	0.9983
WE-MV	0.9922	0.9925	0.9919	0.9983
MLP	0.9925	0.9935	0.9914	0.9983
RF	0.9911	0.9914	0.9909	0.9980
SVM	0.9919	0.9935	0.9903	0.9982
MV	**0.9930**	**0.9941**	**0.9919**	**0.9985**

Results of supervised classifications are depicted at Table-V. We experimented with using only word-embedding based scorings (FT and WV) along with additional features to test the benefit of lexical matching based scores in the supervised setting. We see improvements for all of the classifiers which shows the benefit of lexical matching scoring for news matching.

All supervised methods result in better performance than the best unsupervised score thresholding method (depicted as *COMB. (all)* at Table-V).

The final F1 score is 0.9930 which corresponds to falsely classifying 26 pairs in total. Some of these errors are pairs with the same narratives but for different events, such as lottery results, weather reports, etc. Another misclassification pattern is the huge difference in details between two news articles, such as a document with one sentence v.s. document



(a) JU-text (b) FT-text (c) FT-title (d) CB (Combined)

Fig. 1. Score histograms of negative and positive pairs for some methods and fields

with more than 100 sentences. In these cases, title matching scores are not high enough to balance the text matching scores.

VI. Conclusion

In this paper, we propose novel lexical matching based scorings for matching long text articles of same events with different narratives. Since long articles contain higher number of event related keywords and entities than short text documents such as queries or questions, we showed that, lexical matching based scores are able to obtain a fine discrimination between matched and unmatched pairings, even without the use of labels. We also proposed that obtaining scores for different fields such as title and spot of the news article would make the model more robust. Using these scores as features to be fed to classifiers, together with other length and time based features, improved the performance even further.

As a future work, we plan to work on matching two sets of news as opposed to single news, so that we can benefit from previous matchings. Another effort would be online learning of the model and testing the system in real-time.

References

[1] B. Liu, T. Zhang, D. Niu, J. Lin, K. Lai, and Y. Xu, "Matching long text documents via graph convolutional networks," *arXiv preprint arXiv:1802.07459*, 2018.

[2] Y. Fan, L. Pang, J. Hou, J. Guo, Y. Lan, and X. Cheng, "Matchzoo: A toolkit for deep text matching," *arXiv preprint arXiv:1707.07270*, 2017.

[3] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng, "Text matching as image recognition." in *AAAI*, 2016, pp. 2793–2799.

[4] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Advances in neural information processing systems*, 2014, pp. 2042–2050.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[7] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[8] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[9] C. Corley and R. Mihalcea, "Measuring the semantic similarity of texts," in *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*. Association for Computational Linguistics, 2005, pp. 13–18.

[10] T. Pedersen, S. V. Pakhomov, S. Patwardhan, and C. G. Chute, "Measures of semantic similarity and relatedness in the biomedical domain," *Journal of biomedical informatics*, vol. 40, no. 3, pp. 288–299, 2007.

[11] B. T. McInnes and T. Pedersen, "Evaluating measures of semantic similarity and relatedness to disambiguate terms in biomedical text," *Journal of biomedical informatics*, vol. 46, no. 6, pp. 1116–1124, 2013.

[12] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[13] T. Kenter and M. De Rijke, "Short text similarity with word embeddings," in *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 2015, pp. 1411–1420.

[14] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International Conference on Machine Learning*, 2015, pp. 957–966.

[15] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 259–272, 2016.

[16] S. Pandit, S. Gupta *et al.*, "A comparative study on distance measuring approaches for clustering," *International Journal of Research in Computer Science*, vol. 2, no. 1, pp. 29–31, 2011.

[17] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[18] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[19] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[20] S. Bird, E. Loper, and E. Klein, "Natural language processing with python oreilly media inc," 2009.

[21] A. A. Akın and M. D. Akın, "Zemberek, an open source nlp framework for turkic languages," *Structure*, vol. 10, pp. 1–5, 2007, https://github.com/ahmetaa/zemberek-nlp.

[22] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

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