Ranking Notable News Stories

Extended Abstract

Antonia Saravanou
University of Athens, Athens, Greece
antoniasar@di.uoa.gr

Giorgio Stefanoni
Bloomberg, London, United Kingdom
giorgio.stefanoni@gmail.com

Edgar Meij
Bloomberg, London, United Kingdom
emeij@bloomberg.net

ABSTRACT

The volume of news content being generated all over the world has increased significantly in recent years and systems to process such information in an automated fashion are also becoming increasingly prevalent. One critical component that is missing in such systems, however, is a method to automatically determine how notable a certain news story is. In this paper, we approach the problem of notable news story detection as a ranking task. That is, we use a notable news article as query and rank structured versions of candidate news stories using a learning to rank framework. We employ different combinations of features and gather relevance labels using crowdsourcing. When evaluated, we find that the combination of features in our proposed method improves over using standard ranking methods, and that the trained model generalizes well to unseen news stories.

1 INTRODUCTION

With the recent rise in popularity of all sorts of social media and the resulting increase in citizen journalism, news is increasing in volume and coverage all around the world. As a result, news consumers run the risk of either being overwhelmed due to the sheer amount of news being produced, or missing out on news stories due to heavy filtering. Systems that automatically create structured representations out of news stories for consumption by automated systems for early crisis detection, have the potential to manage the flow of news. These systems typically create structured representations from news articles [6], an example of which is shown in Table 2. The structure that is extracted from each sentence is a so-called triple that represents *who* did *what* to *whom*, associated with *when* and *where* information.

Merely extracting such facts out of unstructured news stories yields enormous amounts of data and we require methods to separate the signal from the noise. One dimension that is typically used in a news context is *notability* which helps identifying those news that are worth reading. In this paper, we address the task of detecting notable news stories structured as triples. We use a trusted source for important news and a stream of candidate triples and aim to answer the question "Which of the candidate triples is notable?". Here, we pick Wikipedia Current Events Portal as the trusted source; that covers international news, trends and developments *of primary importance* on a daily basis. As a proxy to detecting notability directly, we consider each item in the trusted source to be a query and aim to rank the triples in light of that query. One such example is shown in Table 1. In order to rank candidates we apply learning to rank (LTR) and extract features related to each news story, to the pair, and to entities that appear in the pair.

This work was done whilst interning at Bloomberg.

Table 1: Example of a notable news story.

Query
A suicide bomber detonates a vehicle full of explosives at a military camp in Gao, Mali, killing at least 76 people and wounding scores more in Mali's deadliest terrorist attack in history. Date: 17 Jan. 2017
Tagged Query
A [Wikipedia:Suicide_attack] detonates a [Link:Vehicle] full of [Link:Explosive] at a military camp in [Link:Gao], [Link:Mali], [Link:Murder] at least 76 people and wounding scores more in Mali's deadliest [Link:Terrorism] in history. Date: 17 Jan. 2017

Table 2: Example of structured news representations.

Triple			**Metadata**	
Subject	**Predicate**	**Object**	**Date**	**Location**
Armed gang	Carry out suicide bombing	Armed rebel	17 Jan. 2017	Gao, Mali

2 PROBLEM DEFINITION

Let *query* $q_i \in Q$ be a **notable news article** in textual format attached with its publication date. Let c_j be a ***candidate* news story**. Each candidate c_j consists of a *triple* in the form (s, p, o), where s is the subject, p the predicate, and o the object, together with the *location* and the *date* of the event.

Given a query q_i and a stream of candidates C, we aim to rank each candidate $c_j \in C$ by its relevance to the query q_i. A pair $\{q_i, c_j\}$ is considered as *very relevant* when both the query q_i and the candidate c_j are about the same event. For example, the query in Table 1 and the candidate in Table 2 are *very relevant*. A pair is *relevant* when *some* information matches. In all other cases the pair is considered *not relevant*.

3 OUR METHOD

Our method can be summed in 3 steps; first, we create pairs of queries and candidates, then, we extract features from those pairs and, finally, we train a LTR model using the features to rank pairs.

In the first step, we **create all possible pairs** of queries and candidates using two constraints: (1) we use queries and candidates that were published on the same date, and (2) we keep pairs that have at least one word in common.

Next, we **extract features** that can be organized in three categories: (i) features related only to one of the components of the pair (query or candidate), (ii) features related to the pair, and (iii) semantic features. We compute the number of terms in the query and the number of terms in the candidate news story. We calculate the following scoring functions using the original and the stemmed [7] version for the query/candidate in the pair: the Okapi BM25 score

using default settings, TF, and TF-IDF. We further define a similarity score, for the similarity appearance percentage between each combination of triple components and the original query. Finally, we extract **semantic features**. We apply entity linking techniques to tag entities that appear in the candidate news story with the corresponding Wikipedia links/entities and use the TagMe API [3]. An example of an entity-tagged query is shown in Table 1. After tagging both the query and the candidate news article, we calculate the number of common entities and the Jaccard similarity.

In the final step, we train learning to rank models to obtain a ranking of pairs from the most relevant (i.e., the query and the candidate are about the same event) to the least relevant. We show results on the runs using the following LTR algorithms: RankBoost (RB) [4], lambdaMART (LM) [8], and Random Forest (RF) [2]. We experiment using different sets of features. In the baseline features set (B), we include BM25 and TF-IDF scores extracted from the original and the stemmed text of the query and the candidate.

4 DATASETS

For our **notable news articles** we use the Wikipedia Current Events Portal (WCEP) to get a list in chronological order.[1]

For our **structured news candidates** we use the Integrated Crisis Early Warning System (ICEWS) dataset which contains events that are automatically extracted from news articles [1]. The events are triples consisting of coded actions between socio-political actors (a source and a target actor). These actors are coded names that refer to individuals, groups, sectors and nation states. Geographical (city and country) and temporal metadata are also associated with each triple. We show examples of the triples in Table 2.

The Gold Truth Labels. We build a crowdsourcing task to obtain golden truth labels. We used the Figure-Eight crowdsourcing platform and ask annotators to judge the relevance of each pair on a 3-point scale (very relevant, relevant, not relevant).[2] After the preprocess, the dataset is highly skewed, with 3% annotated as *very relevant*, 1% as *relevant*, and 96% is annotated as *not relevant*. In total, the final dataset contains 9.1K pairs; 74 queries (i.e., notable news articles) and the average number of candidate triples per query is 123.35. Finally, to evaluate our method in a real-world setting, we split the dataset by date using the first ten days for training, the next two days for validation, and the last two for testing.

5 RESULTS AND DISCUSSION

In this section we discuss the results of our experiments which are designed to answer the following questions. How well is our method at ranking the query-candidate pairs compared to baselines? How does the performance vary with different parameter settings? Does the small number of *relevant* pairs affect performance? Do we benefit from adding entity-related information?

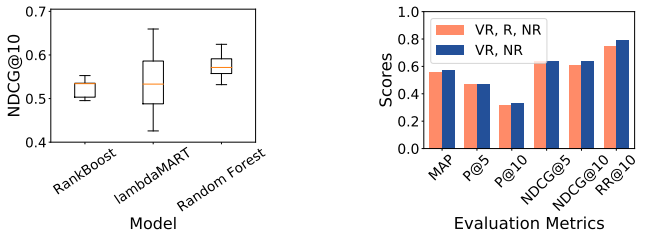
We show the results of the comparison the three LTR models (RB, LM, RF) on the All and B feature sets in Table 3. Our method (using All features) achieves better results than using just the baseline B features (BM25 and TF-IDF). Our method consistently outperforms all baselines, achieving $5-12\%$ improvements on NDCG@10. These improvements are statistically significant with p-value ≤ 0.01.

[1]https://en.wikipedia.org/wiki/Portal:Current_events
[2]https://www.figure-eight.com/

Table 3: Main results of the LTR models on our dataset.

	RB_B	LM_B	RF_B	RB_{All}	LM_{All}	RF_{All}
MAP	0.53	0.44	0.56	0.37	0.34	0.44
P@5	0.42	0.38	0.47	0.33	0.31	0.4
NDCG@5	0.6	0.51	0.64	0.37	0.36	0.42
RR@10	0.75	0.65	0.75	0.6	0.6	0.62

Figure 1: (Left) Results for each model on the validation set for different parameter settings. Each box shows the median (orange line) and the upper/lower quartiles. (Right) Performance using RB with selected features on two datasets.



In figure 1 (Left), we show the mean and standard deviation of the performance using different parameter settings. RankBoost and Random Forest models show less sensitivity in the parameters tuning compared to lambdaMART in our setting. We evaluate the LTR models in ranking pairs using all annotations including the labels: *VR*, *R*, and *NR*. We do the same experiments using only the *VR* and *NR* labeled pairs. As shown in figure 1 (Right), the model achieves better results when excluding the *R* labeled pairs. This was expected as the relevant label is very rare (only 1%) and the model tends to consider it as noise.

6 CONCLUSION AND FUTURE WORK

In conclusion, we presented a method for detecting notable structured news items that leverages textual and semantic features [5]. We show that learning to rank is a viable method to determine notability of news stories. Among the key steps of our method are: (i) the extraction of textual and semantic features, and (ii) the exclusion of the pairs that do no convey strong signal, i.e., the ones labeled as '*relevant*'. The RF model outperforms the rest and it is also more robust with respect to hyperparameter settings. Although our method obtains high performance (RR@10 = 81%), we believe we can attain further improvements by leveraging relations of the identified entities to discover implicitly relevant ones.

REFERENCES

[1] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, and James Starz. 2018. ICEWS Automated Daily Event Data.
[2] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[3] Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). , 1625–1628 pages.
[4] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences.
[5] Antonia Saravanou, Giorgio Stefanoni, and Edgar Meij. 2020. Identifying Notable News Stories. In *European Conference on Information Retrieval*. Springer, 352–358.
[6] Philip A Schrodt. 2001. Automated coding of international event data using sparse parsing techniques.
[7] Cornelis J Van Rijsbergen, Stephen Edward Robertson, and Martin F Porter. 1980. New models in probabilistic information retrieval.
[8] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures.