Background Linking: Joining Entity Linking with Learning to Rank Models

Ornella Irrera[1][0000−0003−2284−5699] and Gianmaria Silvello[1][0000−0003−4970−4554]

Department of Information Engineering, University of Padova
{irreraorne,silvello}@dei.unipd.it

Abstract. The recent years have been characterized by a strong democratization of news production on the web. In this scenario it is rare to find self-contained news articles that provide useful background and context information. The problem of finding information providing context to news articles has been tackled by the Background Linking task of the TREC News Track.
In this paper, we propose a system to address the background linking task. Our system relies on LambdaMART learning to rank algorithm trained on classic textual features and on entity-based features. The idea is that the entities extracted from the documents as well as their relationships provide valuable context to the documents. We analyzed how this idea can be used to improve the effectiveness of (re-)ranking methods for the background linking task.

Keywords: Entity Linking · Graph of Entities · Learning to Rank

1 Introduction

According to Pew Research studies carried out in 2018, the 93% of American adults consume at least some of their news online, via social media recommendations, web browsing or advertising recommendations [14,22]. The adoption of digital strategies marked the end of the publisher-driven news delivery, shifting the focus faraway from the publisher towards the story. In this scenario, the user is allowed to consume news on the web and publish his ones. This had a substantial impact on news production. Indeed, it is more and more challenging to find self-contained news articles and provide context and background information about the story told. The National Institute of Standards and Technology (NIST) recognized to Information Retrieval (IR) and Natural Language Processing (NLP) a primary role in the solution of this problem and, in cooperation with the Washington Post, launched the first edition of the News Track in TREC 2018 [14]. This track is organized into two subtasks, *Background Linking* and *Entity Ranking*: their goal is to provide the user with different means to

understand news articles. The former relies on the construction of a ranked list of articles to provide background information. The latter exploits, as a means of contextualization, a ranking of named entities mentioned in the article the user is reading.

In this paper, we present an IR system based on Learning to Rank methods to improve the ranking of documents for the Background Linking task context.

In the literature, several solutions were proposed to address this task. Most of them treat background linking as an *ad hoc search* task and rely on approaches based on keyword extraction [2,15,26]. Other methods instead, leverage on entities to identify documents' topics [16,17]. In the 2019 edition of the TREC Background Linking task instead, many participants exploited machine learning methods [7], with a particular focus on learning to rank approaches [5,20].

We propose a retrieval system that relies on LambdaMART learning to rank algorithm [3] and the classic BM25 model [21] to rank background articles. In particular, we focus on the creation of feature vectors to be fed to learning to rank methods such as LambdaMART; indeed, we combine features extracted from two different representations of the same document: the unstructured textual representation and a graph-based one. We consider a graph of entities extracted from the textual documents and then linked to Wikipedia articles. In this paper, we analyze advantages and limits of entity-based features in learning to rank approaches and we discuss how they can be employed to improve the final document ranking for the background linking task.

The rest of the article is organized as follows: in Section 2 we provide some background and related work, in Section 3 we describe the key components of the proposed solution and in Section 4 we present a use case. Section 5 describes the experimental setup and Section 6 reports about the evaluation results.

2 Background

The TREC News Track aims to study how to provide contextual information to users while reading news articles. To this end, two tasks are defined: *Entity Ranking* and *Background Linking*. *Background Linking Task* concerns the development of systems able to help users contextualize news articles as they are reading them. Formally, given a source article (i.e., the query), the system should retrieve a list of articles providing relevant background and context information related to the source [14].

The reference collection for this task is the Washington Post Corpus ver.2.[1] This collection contains about 590,000 news articles and blog posts published from 2012 to 2017 by the Washington Post[2]. Each document is characterized by a list of fields such as *id, author, article URL, date of publication, title*. The main content is organized in one or more paragraphs; they may include HTML tags, images and videos. Fifty topics have been provided for this task; each of them is identified by a *number* and by the *id* and the *URL* to the topic's source document

[1] https://trec.nist.gov/data/wapost/
[2] https://www.washingtonpost.com/

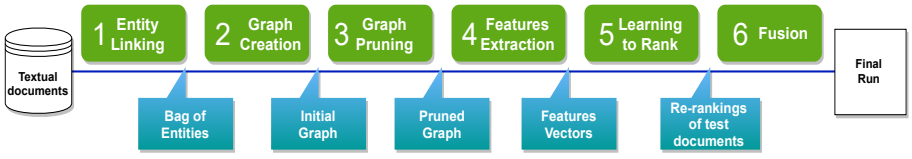

Fig. 1: Representation of the main phases composing the proposed solution. Green rectangles show the six phases, while the blue ones show the output of each phase.

(or *query*). Graded relevance judgments are ranging from 0 (not relevant) to 4 (recommended article).

In the proposed solution, we leverage document graph representation based on the entities and their relationships that we automatically extract from the documents. In the literature, several approaches take advantage of entity-oriented representations. Xiong et al. in [24,25] propose a Bag-of-Entities (BoE) model where each document is represented as a bag-of-entities constructed via entity linking. The ranking of documents is generated, considering the overlapping entities between each document and the query. A similar approach is [11] that describes a learning to rank approach where the training is based both on Bag-of-Words (BoW) and BoE features. The experimental evaluation showed that the combination of features depending on words and entities improve a BM25 baseline. What makes our retrieval system different from the solutions proposed above, is that we take the separated entities belonging to the BoE constructed via entity linking and we create a graph; this graph is more informative than the classic BoE representation because it includes both the information related to the separated entities and the information carried by their relationships.

3 Proposed Solution

The main phases composing the proposed solution are reported in Figure 1. In the following, we describe every single phase.

Entity Linking. In this phase, we perform entity recognition on the textual documents to extract a set of mentions to be linked with entities in a knowledge repository (KR) [1] – i.e., Wikipedia[3] in our case. This process can be called entity *annotation*. This phase's output is a *bag-of-entities* for each document.

Graph Creation. This phase takes the bag-of-entities as input and creates an undirected weighted graph, where the nodes are the entities and the edges are based on the *semantic relatedness* between the nodes. The *semantic relatedness* is a measure defined in [23], which exploits the Wikipedia structure to find the

[3] https://www.wikipedia.org/

relatedness between two entities. Two entities are semantically related if they share a high number of entities linking to them [1]. In our implementation, the entity pairs are connected by an edge whose weight is the numerical value of the semantic relatedness. The output of this phase is a graph with one or more connected components. There could be a strong imbalance between the different components; in fact, one may include the largest part of the nodes, while another contains very few entities. It is common to identify in the largest connected component one or more *communities* – or groups of nodes highly connected within themselves and poorly with the other groups [4].

Graph Pruning. In this phase we remove the meaningless entities from the graph. Pruning is based on *components removal*, which keeps only the largest connected component of the graph and *community detection* that detects the largest community, where the most representative entities usually are. To this end, we rely on the Girvan-Neman algorithm for community detection [10] based on the "edge betweenness", a generalized version of the classic betweenness centrality measure defined in [8]. The edge betweenness of an edge corresponds to the number of shortest paths between every pair of vertices that run along it [10]. Removing some nodes from a graph may cause a loss of information, especially if two large connected components coexist in the same graph (in this case the separation between different components shows that the article discusses two weakly correlated subjects). Since this case is extremely rare, the advantages brought by the pruning phase predominate over the possible loss of information.

Features Extraction. This phase is about the definition and extraction of the features representing the documents; there are document-based and query-based features. The two feature sets comprise both textual and entity-based graph features. Since the overall goal of the implementation consists in studying the impact of the graphs of entities in the learning process, the core of this phase lays in the extraction of the entity-based graph features. The query graph-based features are the most informative about the relevance of a document because they reflect the similarities between the document graph and the query one. This is why the largest part of the extracted features belong to this type. An example of query graph-based features is the semantic relatedness [23] between the most central node of each graph, where the centrality measure considered is the betweenness centrality [8]. High values highlight the correlation between the document graph and the query one. The features computed independently of the query graph, instead, are topological properties of the document graph. This type of features is less informative than the previous one because it does not highlight any relation of the document with the query; however properties such as the node connectivity and the node's degree revealed their usefulness in relevance identification. Only few features belong to the textual-based type, and most of them depend on the query article. Some examples of query-based textual features are the BM25 score and the Term Frequency (TF), while the document's length and the number of paragraphs are document-based textual features. In Table 1 it is presented an overview of the number of features extracted from

Table 1: Overview of the features extracted.

Type	Subtype	# of features	Total
Textual-based features	Document-based	3	10
	Query-based	7	
Entity-based graph features	Document graph-based	12	55
	Query graph-based	43	

each document's representation. For each document it is finally created a feature vector. These vectors are required to perform learning to rank tasks.

Learning to Rank. We employ LambdaMART because it is one of the best performing learning to rank methods [3] in the literature. LambdaMART is a list-wise approach based on Multiple Additive Regression Trees (MART) [9]. LambdaMART is trained to automatically construct a set of ranking models. The ranking models trained on different sets of hyper-parameters are tested on the test set containing a new list of queries; the documents in the test set are ranked on the base of the new scores assigned by the models. The output of this phase is the set of final runs, one for each model.

Fusion. In this phase we experiment the fusion of multiple system's runs to analyze whether an increase in the number of merged runs corresponds to an effectiveness improvement. Each run provided by the learning to rank phase contains the same set of documents ranked differently. In order to create the fusion runs we employ *combSUM* method [18]: the new final score of a document corresponds to the sum of the scores that document obtained in each individual run. The documents are finally re-ranked according to their new scores.

4 Use case

To better illustrate how a graph of entities is created starting from a textual article, we employ a small portion of an article[4] about tropical storms.

Entity Linking. The mentions are detected and linked to Wikipedia entities. Below, we report the textual fragment where the mentions are marked in boldface.

> **Super Typhoon** Vongfong **explodes** becomes most intense **storm** on **Earth** in 2014. Super **Typhoon Vongfong** has rapidly intensified over the **past 24 hours** from the equivalent of a **category two hurricane** to a monster **typhoon** [...]

[4] Article available at: https://www.washingtonpost.com/news/capital-weather-gang/wp/2014/10/07/super-typhoon-vongfong-explodes-becomes-most-intense-storm-on-earth-in-2014/

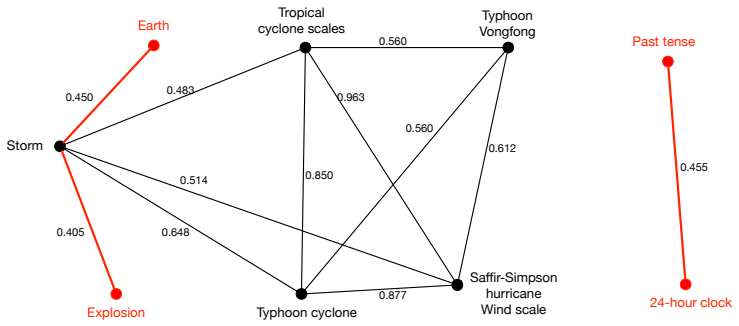

Fig. 2: An example of entity graph. In red the nodes and edges removed after the graph pruning phase.

In Table 2 we report the mentions and their linked entities. Giving a first look to the mention-entity associations, the largest part of entities are coherent with the tropical storms except for *past tense* and *24-hour clock* which are very general or out of context.

Table 2: Mention - Wikipedia entity associations

Mention	**Wikipedia entity**
Super Typhoon	Tropical cyclone scales
explodes	Explosion
storm	Storm
Earth	Earth
Typhoon Vongfong	Typhoon Vongfong (2014)
past	Past tense
24 hours	24-hour clock
category two hurricane	SaffirSimpson hurricane wind scale
typhoon	Tropical cyclone

Graph Creation. The extracted entities are the nodes of the graph, while the edges are based on the semantic relatedness between the nodes. In Figure 2 we show the resulting graph. The two connected components are highly unbalanced because a component includes the largest part of entities while the second has only two entities. In the largest connected component lie the most meaningful entities, while the smallest one contains the two out of context entities, *past tense* and *24-hour clock*.

Graph Pruning. Starting from the graph in Figure 2, we firstly remove the smallest connected component (right part in red). Then, we run the Girvan-Newman algorithm that removes the *Earth* and *Explosion* entities. These entities

are discarded because they have degree equal to one and they do not belong to the largest community. Nonetheless, if we consider these entities in the context of tropical storms, they do not bring any contribution for context identification, hence their removal increases the coherence of the graph with respect the topic.

5 Experimental Setup

In this section, we describe the experimental setup. In particular, we discuss some technical details, and we illustrate the tools used to implement each phase of the proposed solution.

Preprocessing. We preprocessed the Washington Post Corpus to remove from each article all the fields without informative content. Then we indexed the collection with Apache Solr[5], an open-source search library based on Apache Lucene set with the default English tokenizer, the English stop-words list and no stemmer. Once indexed the collection, we used the default BM25 in Solr to construct an initial ranking of 100 documents per topic: our **baseline**.

Entity Linking and Graph creation. Since it was unfeasible to analyze the entire collection of documents in order to choose the most effective settings' apporach, we relied on a sample of thirty documents, both relevant and not relevant, and we considered the setting the most effective on this sample. The linking system we adopted is TagMe [6], in particular, we performed article annotation via the TagMe RESTful API, setting a confidence score $\rho = 0.1$. This parameter is a threshold imposed to discard non-meaningful entities [13]: the lower this parameter is, the more entities are recognized. For each article, we linked at most thirty entities leading to graphs with at most thirty nodes. To create graphs as consistent as possible with the original textual article, we set a threshold equal to 0.4 on the value of semantic relatedness. Specifically, if two entities have a semantic relatedness higher (or equivalent) than 0.4, an edge between them is created. This threshold allowed us to obtain graphs subdivided into connected components: such a structure highlights the distinction between meaningful and non-meaningful entities. Pruning is performed only if the graph contains at least ten nodes: if it does not, pruning will lead to a graph unable to represent the original document correctly. To perform graph creation and pruning, we relied on NetworkX Python library [12].

Features Extraction. To extract the set of textual-based features from each document, we exploited Scikit-learn [19], a Python library that provides the implementation of measures like the TF and the Inverse Document Frequency (IDF). For the extraction of the entity-based graph features, both document-based and query-based, we exploited the NetworkX Python library [12]. We created for each document a feature vector containing the following elements: the relevance

[5] https://lucene.apache.org/solr/

Table 3: Overview of training and test sets.

Set	Track	Collection	Topics	# of Vectors	Total
Training set	News Track 2019 Background Linking task	Washington Post Corpus vs2	57	8127	20863
	Common Core Track 2017	New York Times Annotated Corpus	44	12736	
Test set	News Track 2018 Background Linking task	Washington Post Corpus vs2	50	5000	5000

judgment of the document, the id of the query, the features extracted and the id of the document.

Learning to Rank. We relied on the implementation of LambdaMART provided by RankLib[6], a library of learning to rank algorithms developed by the Lemur Project. We performed manual tuning, which allowed us to understand the best combinations of hyperparameters and analyze how each parameter interferes with the others. Table 3 summarizes the main features related to the training and test sets involved in our implementation. The test set comprises the vector representation of the documents belonging to the baseline. The training set, instead, combines two sets of topics belonging to different TREC tracks. We remark that even if both the training and test sets contain documents belonging to the Washington Post Corpus, they rely on disjointed sets of topics. Performing training depending on two different collections of documents allowed us to enrich the training set but, at the same time, it made our system suffer from the limitations induced by the *transfer learning*. In particular, what influenced our system the most was the lack of consistency between the relevance grades provided for the two tracks; indeed, we had to map the five possible relevance grades offered for the Background Linking task to the three grades provided for the Common Core Track of TREC 2017. The training set was finally split to derive a validation set. We trained the algorithm by choosing more than 200 different combinations of hyperparameters, and we selected the seventy models which maximized the nDCG@5 effectiveness on the validation set, and, at the same time, which were far from overfitting or underfitting conditions. In Table 4, the values of the five most effective models' hyperparameters are described. We tested the seventy ranking models on the test set, and we obtained seventy final runs: each one contains a different re-ranking of the baseline documents.

Fusion. To perform fusion we considered the ten most effective models and the related final runs. We fused k (with $k = \{2, 3, 5, 7, 10\}$) runs, and, for each k, we collected $\binom{10}{k}$ runs – i.e. all the possible combinations of k distinct runs taken from a set of ten runs.

[6] https://sourceforge.net/p/lemur/wiki/RankLib/

Table 4: Hyperparameters' combinations of the six best models. In bold text we marked the combination related to the model whose nDCG@5 was the highest.

Trees	Leaves	Shrinkage	Threshold candidates
600	6	0.08	all
1000	7	0.06	256
2000	5	0.05	all
2000	8	0.03	256
2500	5	0.01	all
2500	**8**	**0.01**	**256**

Evaluation and metrics. Coherently with the guidelines of the task [14], we considered nDCG@5 as primary measure of effectiveness and we tested the system's performances also in terms of nDCG@1, nDCG@10, nDCG@100, Precision at Cut-off 1 (P@1) and reciprocal rank (recip_rank). We analyzed two types of runs: (1) the *individual run*, intended as the re-ranking produced by each one of the seventy LambdaMART's models, and (2) the *fused run*, intended as the result of the fusion of k individual runs belonging to the set of ten runs dedicated to the fusion approach. We used trec_eval[7] tool to evaluate the set of seventy individual runs and the five sets of fused runs (one set of fused runs for each k); for each set we selected the run with the highest nDCG@5, obtaining in total six selected runs.

6 Evaluation and results

In this section we describe the results obtained by our system. We assess the retrieval effectiveness from both a *quantitative* and a *qualitative* viewpoint.

6.1 Quantitative evaluation

The quantitative evaluation has the intent to describe how our system performs on average. To this end, in Table 5 and Table 6 we illustrate the average results of the six selected runs and the baseline. We indicated the baseline as *BM25*, the best individual run as *best run* and the five best runs obtained with the fusion approach as *fus* followed by the number of runs fused. The top scores of each evaluation measure are highlighted in boldface.

In Table 5 we describe the performances of our system in terms of reciprocal rank, P@1 and nDCG@1: these metrics evaluate the effectiveness of one document per topic in a run. Our goal is to evaluate if our system can correctly recognize a relevant document and place it in the ranking's highest positions. In general, all the generated runs show effectiveness slightly higher than the baseline both for reciprocal rank and nDCG@1. In P@1 instead, three runs equalled

[7] https://trec.nist.gov/trec_eval/

Table 5: Evaluation results of one document per topic.

	recip_rank	P@1	nDCG@1
best run	0.8128	0.7400	**0.4150**
fus2	0.8269	0.7400	0.3825
fus3	0.8378	0.7600	0.4100
fus5	**0.8473**	**0.7800**	0.4100
fus7	0.8373	0.7600	**0.4150**
fus10	0.8240	0.7400	0.3950
BM25	0.7951	0.7400	0.3525

Table 6: Evaluation results at different cut-offs.

	nDCG@5	nDCG@10	nDCG@100
best run	0.4090	0.4055	0.4589
fus2	**0.4168**	0.4100	0.4601
fus3	0.4166	0.4106	0.4635
fus5	0.4129	0.4131	0.4646
fus7	0.4139	0.4038	0.4636
fus10	0.4017	0.4077	0.4602
BM25	0.4097	**0.4233**	**0.4659**

the baseline (*best run, fus2* and *fus10*) and three runs reported greater values (*fus3, fus5* and *fus7*). The most effective run is *fus5*, obtained fusing five individual runs. It maximizes both reciprocal rank and P@1 evaluation measures. *best run* and *fus7* instead, maximize the nDCG@1. Observing the reported results, we can notice that increasing the number of runs fused does not necessarily correspond to an effectiveness improvement. The effectiveness of *fus5* and *fus7* runs in fact, is usually higher than the effectiveness of *fus2*, *fus3* and *fus10*: this indicates that the fusion can improve the performances until a certain number of runs fused, after that, the fusion becomes disadvantageous and the effectiveness decreases. This is proved by *fus10* which is the least effective run among the five fused runs. This behavior is verified in all the measures proposed in Table 5.

Table 6 reports the effectiveness of the proposed runs at different ranking depths. This analysis considers the nDCG evaluation measure at different cut-offs: 5, 10 and 100 (in this case, it is evaluated the entire ranking associated with each topic). In this analysis, we study our system's ability to produce effective rankings. In particular, we are interested in how the effectiveness varies, increasing the cut-off. The results reported in terms of nDCG@5 highlight that the *fus2, fus3, fus5* and *fus7* runs have performances slightly above those of the baseline, while the *best run* approximates it very well. The least effective run in nDCG@5 is the *fus10* which achieves the lowest result. If we consider more than five documents in the ranking, the BM25 model always reports the

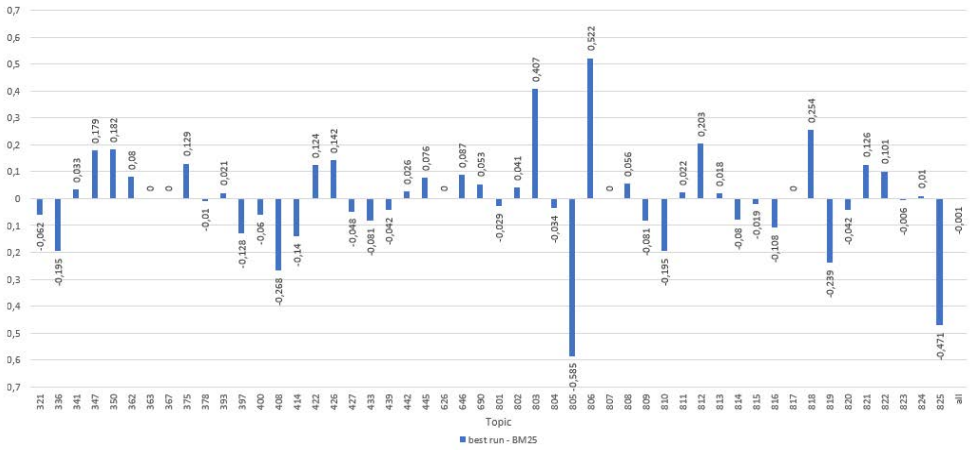

Fig. 3: This histogram represents the nDCG@5 effectiveness of the *best run* compared to the nDCG@5 of the baseline. Each column represents a topic and the column's height is found computing the difference between the nDCG@5 of the *best run* and the nDCG@5 of the baseline for that topic.

highest effectiveness. The highest results achieved by our system for nDCG@10 and nDCG@100 are those of *fus5*, while the lowest ones are attributed to the *best run* and *fus10*. As it has been already noticed in Table 5, this revealed that the fusion approach improves the system's effectiveness, but it becomes disadvantageous if more than five runs are fused.

Discussion First, our hypothesis that fusing more runs leads to an effectiveness improvement does not seem to be supported. Both in Table 5 and Table 6 the fusion of all the available runs is the least effective run among the six proposed. We also see that there is no correlation between the number of runs fused and the effectiveness improvement. Despite this fact, the fusion approach has been revealed to be useful in our implementation, since among the selected runs, the fused ones achieve the highest results. Moreover, our system and the BM25 model show two opposite behaviors: the more documents we consider in the ranking, the more our system's effectiveness decreases, and vice versa holds for the BM25 model. This contrast between our system and BM25 is primarily caused by the transfer learning. In fact, the effectiveness decrease detected after only five documents is strictly related to the inability of our system to distinguish among more than three relevance scores.

6.2 Qualitative evaluation

The qualitative evaluation describes the effectiveness of the particular topics belonging to a run. The histogram shown in Figure 3 describes the difference

between the nDCG@5 score of the *best run* for a given topic and that one of the baseline. In particular, all the columns above the abscissa identify the topics where our system prevails over the baseline. Vice versa holds for the columns below the abscissa. It is possible to notice an equivalent number of topics laying above and below the abscissa; this means that the effectiveness improvement brought by the topics where our system prevails over the baseline is perfectly balanced by the effectiveness decrease caused by the topics where BM25 prevails over our system. This is the reason why, in the quantitative analysis, the nDCG@5 of the *best run* (0.4090) approximated the one of the baseline (0.4097). We conducted an analysis focused on the rankings associated with the best and the worst performing topics of the *best run* – this study aimed at finding a correlation between the documents' feature vectors and our system's effectiveness. The most compelling topics report in the first positions of their rankings documents whose relevance is identifiable both in the textual-based and in the entity-based graph features. This proves that relying on features that leverage the classic textual representation and the graph of entities can lead to an effectiveness improvement. Analyzing the rankings associated with the topics where the BM25 prevails, we identified three main reasons for our system's poor effectiveness in some topics. These are:

- The transfer learning: it prevented some topics from reaching a high level of effectiveness.
- The graphs of entities: the documents may not present well-formed graphs; in this case, the learner attributes a score on the basis of the textual features only. This happens when the original article has not enough textual content to construct a consistent graph.
- Errors in learning phase: this condition depends on the features in the vectors; we experimented that the value of a single feature may influence the entire ranking of a topic.

These conditions are unavoidable, and they represent the cases in which it is convenient to use the BM25 model.

7 Conclusions

This article presented a solution for the background linking task relying on LambdaMART to obtain a list of background articles. We leveraged the document's textual and graph representations to extract a set of features used to perform training. Our goal was to study whether the combination of features belonging to different document representations can improve the system's effectiveness; in particular, we were interested in exploring the impact related to the graphs of entities-based features. The analysis conducted on the single topics highlighted that there is a set of topics where our system outperformed the BM25 model. In these cases, the graphs of entities played a crucial role because the combination of textual-based features and graph-based ones allowed for an effectiveness improvement. This implied that our initial hypothesis is confirmed

for this first set of topics. However, there was an equivalent number of topics where our system was not highly effective. Transfer learning has a high impact on negative performances. The balance between the effective and ineffective topics explained the similarity between the average nDCG@5 values obtained by our system and by the baseline. We finally introduced the fusion approach as a means to improve the overall effectiveness of our system. The results showed that, contrary to our initial belief, fusing too many runs makes the performances decrease; in particular, the optimal number of runs to merge, in the tested context, is five.

References

1. Balog, K.: Entity-oriented search. Springer Nature (2018)
2. Bimantara, A., Blau, M., Engelhardt, K., Gerwert, J., Gottschalk, T., Lukosz, P., Piri, S., Shaft, N.S., Berberich, K.: htw saar@ trec 2018 news track. In: TREC (2018)
3. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
4. Despalatović, L., Vojković, T., Vukicevic, D.: Community structure in networks: Girvan-newman algorithm improvement. In: 2014 37th international convention on information and communication technology, electronics and microelectronics (MIPRO). pp. 997–1002. IEEE (2014)
5. Ding, Y., Lian, X., Zhou, H., Liu, Z., Ding, H., Hou, Z.: Ictnet at trec 2019 news track. In: TREC (2019)
6. Ferragina, P., Scaiella, U.: Fast and accurate annotation of short texts with wikipedia pages. IEEE software **29**(1), 70–75 (2011)
7. Foley, J., Montoly, A., Pena, M.: Smith at trec2019: Learning to rank background articles with poetry categories and keyphrase extraction. In: TREC (2019)
8. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry pp. 35–41 (1977)
9. Friedman, J.H., Meulman, J.J.: Multiple additive regression trees with application in epidemiology. Statistics in medicine **22**(9), 1365–1381 (2003)
10. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
11. Gonçalves, G., Magalhães, J., Xiong, C., Callan, J.: Improving ad hoc retrieval with bag of entities. image **409**(68.81), 116 (2018)
12. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008)
13. Hasibi, F., Balog, K., Bratsberg, S.E.: On the reproducibility of the tagme entity linking system. In: European Conference on Information Retrieval. pp. 436–449. Springer (2016)
14. Huang, S., Soboroff, I., Harman, D.: Trec 2018 news track. NewsIR@ ECIR **2079**, 57–59 (2018)
15. Lu, K., Fang, H.: Paragraph as lead-finding background documents for news articles. In: TREC (2018)
16. Lu, K., Fang, H.: Leveraging entities in background document retrieval for news articles. In: TREC (2019)

17. Missaoui, S., MacFarlane, A., Makri, S., Gutierrez-Lopez, M.: Dminr at trec news track. In: TREC (2019)
18. Montague, M., Aslam, J.A.: Relevance score normalization for metasearch. In: Proceedings of the tenth international conference on Information and knowledge management. pp. 427–433 (2001)
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
20. Qu, J., Wang, Y.: Unc sils at trec 2019 news track. In: TREC (2019)
21. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Now Publishers Inc (2009)
22. Soboroff, I., Huang, S., Harman, D.: Trec 2018 news track overview. In: TREC (2018)
23. Witten, I.H., Milne, D.N.: An effective, low-cost measure of semantic relatedness obtained from wikipedia links (2008)
24. Xiong, C., Callan, J., Liu, T.Y.: Bag-of-entities representation for ranking. In: Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval. pp. 181–184 (2016)
25. Xiong, C., Callan, J., Liu, T.Y.: Word-entity duet representations for document ranking. In: Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval. pp. 763–772 (2017)
26. Yang, P., Lin, J.: Anserini at trec 2018: Centre, common core, and news tracks. In: Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC 2018), Gaithersburg, MD (2018)