IRIT at TREC 2019: Incident Streams and Complex Answer Retrieval Tracks

Alexis Dusart, Gilles Hubert, and Karen Pinel-Sauvagnat

{alexis.dusart, gilles.hubert, karen.sauvagnat}@irit.fr,
Universite de Toulouse UPS-IRIT,
118 route de Narbonne F- 31062 Toulouse cedex 9

Abstract. This paper presents the approaches proposed by the IRIS team of the IRIT laboratory for the TREC Incident Streams and Complex Answer Retrieval tracks. The Incident Streams (IS) track aims to categorize and prioritize tweets in disaster situation to assist emergency service operators. In our participation, we applied supervised learning techniques based on features extracted from tweets.
Then, the 2019 edition of the Complex Answer Retrieval (CAR) track aims to answer complex questions expressed as outlines using paragraphs from Wikipedia. In our participation, we used the Terrier retrieval system to rank paragraphs for each section of the outline and keep the most relevant according to three different strategies.

1 Introduction

In 2019, we participated in both the Incident Streams and Complex Answer Retrieval tracks. As these tracks are completely independent, we separately present our approaches for the two tracks in section 2 and 3.

2 Incident Streams

This year, the track was organized in two different parts, both having the same objective. Part A ran in June and Part B in September. The training test of Part B included the test set of Part A.

2.1 Overview of the TREC IS task

Today, social media is full of information, including people trying to contact emergency services in crisis situations [1].

The purpose of the TREC Incident Streams track is to use this social information to help emergency services, as shown in Figure 1 [2]. Tweets have to be classified according to predefined categories called Information types, as for instance "Information wanted" or "Request for Search and Rescue". The task also consists in giving a priority score to the tweets in order to return alerts if necessary. An alert is raised when the priority score is greater than 0.7. Particular attention is paid to information types named "Requests for Goods/Services", "Requests for Search and Rescue", "Calls to Action for Moving People", "Reports of Emerging Threats", "Reports of Significant Event Changes" and "Reports of Services" that are considered actionable, which means that they should lead to actions from emergency services.

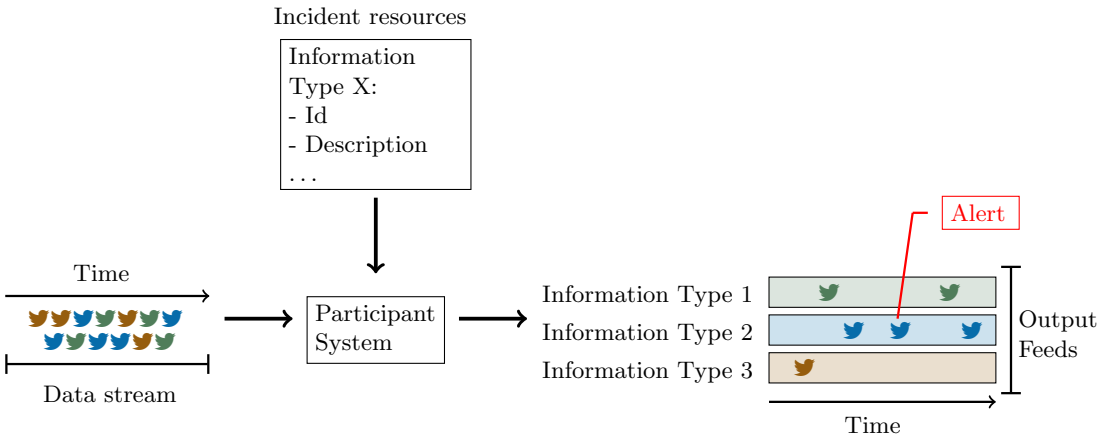

Fig. 1. Incident Streams task, as shown on the task's website (`http://dcs.gla.ac.uk/~richardm/TREC_IS/`)

2.2 Part A

As aforementioned, the Part A of the task ran in June 2019. The number of learning tweets was about 18,000 and the number of test tweets was about 7,000. In this section we present our system framework, the results, and the corresponding failure analysis.

2.2.1 System framework

Features extraction

In order to identify the different categories and priorities, we extracted some features related to tweets, which we categorized as follows. *Community features* show the impact of a given tweet in the Twitter community. *Morphologic features* aim at distinguishing between different non-verbal ways of conveying a message. *Microblogging specific features* consider the specific content of tweets such as emoticons or hashtags. *Grammatical features* convey the grammatical quality of a tweet. *Entities features* are used to find persons, organizations, and locations. Finally, the goal of *sentiment features* is to represent possible sentiments expressed in the tweet. These features are described in details below:

- *Community features*: number of retweets of the tweet and, for the tweet's author, number of friends, number of followers, number of statuses;
- *Morphologic features*: number of upper case characters and upper case words, number of stopwords, number of exclamation and question marks, maximum number of consecutive exclamation and question marks, length of the tweet content (number of words), average word length of tweet content (number of characters);
- *Microblogging specific features*: presence and number of emoticons, presence and number of hashtags, presence and number of URLs, number of phone numbers;

- *Grammatical features*: number of nouns, verbs, gerund verbs, adverbs, pronouns, WH question words in the content of the tweet;
- *Entities features*: number of locations, persons and organizations;
- *Sentiment features*: presence of positive, negative and compassion sentiment.

Emojis, emoticons, URLs, hashtags, phone numbers, upper case, exclamation, and question marks features are extracted before any text cleaning. For the other features, we applied a lowercase transformation, punctuation removal, and stopword removal using the nltk library [3]. The length of tweet content and number of stopwords features are extracted before stopwords removal.

For the sentiment features, we used pre-trained weights of DeepMoji [4], as used in the architecture presented in [5] for the 2018 Incident Streams task. More precisely, we retrieved the 5 emojis that DeepMoji predicted for each tweet. We manually associated to each emoji one or several sentiment polarities among positive, negative, and compassion. If the majority of the emojis convey positive, negative or compassion sentiments, the tweet is labelled with the corresponding feeling.

For entities and grammatical features, we respectively used Stanford Named Entity Recognizer [6] and Stanford Part-Of-Speech Tagger [7]. Finally, the other features are extracted using regular expressions.

Information Type prediction

From the extracted features, we predicted the information types in the tweets using supervised learning algorithms. Inspired by the work in [5], we combined two models of supervised learning. The combination that gave us the best results in the validation phase was Random Forest Classifier [8] and Gradient Tree Boosting Classifier [9].

In the task, a tweet can be associated with one or more information types. To tackle this multi-label problem, we proposed three different strategies : (i) Binary Relevance[1], (ii) a fixed number of information types to return, (iii) a class membership threshold.

Priority score evaluation

Each tweet in the training set was associated with an importance label among Low, Medium, High, and Critical. These labels are respectively associated to priority scores of 0.25, 0.5, 0.75, and 1.

To predict the priority scores in the test set, we performed a linear regression using the previously described features.

2.2.2 Runs

Our runs for Part A were named `IRIT_run_rf_gb`, `IRIT_run_rf_gb_binary` for the official submissions, `IRIT_run_rf_gb_threshold` and are respectively renamed `A_Fixed`, `A_Binary_Relevance` and `A_Threshold` in this notebook to be more expressive. The associated configurations are detailed in Tables 3 and 4.

[1] `http://scikit.ml/api/skmultilearn.problem_transform.br.html`, last access on October 22, 2019.

For run `A_Binary_Relevance`, we discretized the continuous features in order to improve our predictions, as done in [10] and [11]. Regarding the multi-label problem and as explained before, runs `A_Binary_Relevance`, `A_Fixed`, `A_Threshold` respectively implement the following strategies: (i) Binary Relevance, (ii) a fixed number (experimentally fixed to 5 according to preliminary experiments) of information types to return, (iii) a class membership threshold (experimentally fixed to 0.06).

2.2.3 Results

The results for Part A are presented in Table 1. For Alerting metrics, the scores are between -1 and 1, 1 being the best score and -1 being the worst. Regarding now Information feed metrics, The higher the score is, the better it is. On the contrary, for priorization scores, the closer the score is to 0, the better it is.

We can notice that we were ranked first regarding Priorization metrics.

Run	Alerting		Information Feed			Priorization	
	Accumulated Alert Worth		Info. Type Positive F1		Info. Type Accuracy	Priority MSE	
	High Priority	All	Actionable	All	All	Actionable	All
A_Fixed	-0.9867	-0.4935	0.0000	0.1735	0.8209	0.1170	0.0558
A_Binary_Relevance	-0.9942	-0.4971	0.0387	0.1716	0.8780	0.0751	0.0694
A_Threshold	-0.9867	-0.4935	0.0117	0.1774	0.7840	0.1170	0.0558
Median Scores	-0.9680	-0.4869	0.0536	0.1622	0.8581	0.1615	0.0756
Best Scores	-0.1213	-0.1839	0.1969	0.2512	0.8829	0.0751	0.0558

Table 1. TREC IS 2019 Part A Results. Runs `A_Fixed`, `A_Binary_Relevance` and `A_Threshold` respectively correspond to the official runs `IRIT_run_rf_gb`, `IRIT_run_rf_gb_binary`, `IRIT_run_rf_gb_threshold`.

2.2.4 Failure Analysis

Regarding the results, it can be seen that the prediction of the actionable information types is worse than the mean prediction for all information types (0.0387 for the best run vs 0.1716 for the worst run).

A detailed analysis led us to see that the information types that have fewer observations in the learning sample get lower F1 scores in the test set. We therefore hypothesized that to improve our predictions we need more data on these information types.

We can also see from the results that the priority scores of our runs are close to the best scores, while the alerting scores are close to the worst. Our failure analysis also revealed that very few alerts were returned by our model (a maximum of 14 out of 345 in the ground truth). We recall that an alert is raised when the priority score is greater than 0.7. An hypothesis is that using a linear regression to evaluate the priority scores does not effectively discriminate between alerting tweets and others.

2.3 Part B

For this edition 2019-B run in September, the number of learning and test tweets was respectively about 25,000 and 14,000.

Our runs for Part B were named `IRITrun1`, `IRITrun2`, `IRITrun3`, `IRITrun4` fot hte official submissions and are respectively renamed `B_R_Resampled`, `B_Binary_Relevance`, `B_T_Resampled` and `B_Threshold` in this notebook to be more expressive. Tables 3 and 4 sum up the methods used for the different runs to classify the different information types as well as to predict priority scores.

The differences in our runs between Part A and Part B are the following:

- We used the POS-Tagger and NER from nltk[2] which are faster than Stanford's ones.
- As proposed by the University College Dublin team in Part A of the task, we used SMOTE [12] to deal with the imbalanced types of information in the training set. SMOTE is a technique for adding observations to under-represented classes in a dataset.
- To evaluate priority scores, unlike linear regression where we considered the variable to be explained quantitatively, we considered the variable to be explained qualitatively among the values Low, Medium, High, and Critical. For this purpose, we used Random Forest that gave us the best results in the validation phase.

The best run results for each metric as well as our scores for Part B are presented in Table 2.

We tried resampling (SMOTE) for the imbalanced types of information and especially for actionable types of information without improving results. Random Forest gives us better results than linear regression to detect high priority tweets.

Run	Alerting		Information Feed			Priorization	
	Accumulated Alert Worth		Info. Type Positive F1		Info. Type Accuracy	Priority MSE	
	High Priority	All	Actionable	All	All	Actionable	All
Median Scores	-0.9197	-0.4609	0.0386	0.1055	0.8583	0.1767	0.1028
Informedia-nb	-0.9197	-0.4609	0.1321	0.0995	0.8605	**0.0788**	**0.0544**
Informedia-rf3	**-0.0898**	**-0.1837**	0.0592	0.0568	0.8434	0.1660	0.2063
B_Threshold	-0.9744	-0.4872	0.0000	0.1583	0.7576	0.1461	0.0775
B_T_Resampled	-0.8482	-0.4351	0.0000	0.1388	0.8565	0.1771	0.1028
B_Binary_Relevance	-0.9794	-0.4897	0.0248	0.1734	0.8534	0.1175	0.0659
B_B_R_Resampled	-0.5649	-0.3332	0.0151	**0.1863**	0.8418	0.1316	0.0911
nyu.fast.multi	-0.9287	-0.4679	0.0854	0.1253	**0.8808**	0.2153	0.1185
UCDbaseline	-0.7856	-0.4131	**0.1355**	0.1343	0.7495	0.0859	0.0668

Table 2. TREC IS 2019 Part B Results. Runs `B_B_R_Resampled`, `B_Binary_Relevance`, `B_T_Resampled` and `B_Threshold` respectively correspond to the official runs `IRITrun1`, `IRITrun2`, `IRITrun3`, `IRITrun4`.

[2] `https://www.nltk.org/book/ch07.html`, last access on October 22, 2019.

Run	Part A	Part B	Threshold	Fixed	B.R.	Discrete	SMOTE
A_Threshold	√		√				
A_Fixed	√			√			
A_Binary_Relevance	√				√	√	
B_Threshold		√	√				
B_Threshold_Resampled		√	√				√
B_Binary_Relevance		√			√	√	
B_Binary_Relevance_Resampled		√			√	√	√

Table 3. Runs Incident Streams Overview Incident Type Classification, B.R. is for Binary Relevance and Discrete for discretized features.

Run	Part A	Part B	Linear Regression	Random Forest
A_Threshold	√		√	
A_Fixed	√		√	
A_Binary_Relevance	√		√	
B_Threshold		√	√	
B_Threshold_Resampled		√		√
B_Binary_Relevance		√	√	
B_Binary_Relevance_Resampled		√		√

Table 4. Runs Incident Streams Overview Priority Score Prediction.

3 Complex Answer Retrieval

3.1 Overview of the TREC CAR track

Current retrieval systems provide good solutions towards phrase-level retrieval for simple fact and entity-centric needs. The Complex Answer Retrieval track aims to return complex information in a structured form, i.e., to answer open questions requiring several pieces of information.

The problem of the CAR 2019 task is to fill outlines structured in sections and given as queries with ordered paragraphs. For each outline, a maximum of 20 paragraphs should be returned, each related to one of the sections of the outline. The provided data are thus the outlines and the corpus of possible paragraphs to be returned:

- The outlines are based on the Textbook Question Answering dataset[3]. The Textbook Question Answering dataset includes college-level lessons in the life, earth and physics sciences [13]. There are 132 outlines, 726 sections of outlines, i.e., an average of 5.5 sections per outline.
- The paragraphs to be returned are provided from Wikipedia, the collection has 29,794,697 paragraphs.

3.2 System framework

Figure 2 sums up our approach. We first indexed the corpus of paragraphs provided to answer the outlines. We then defined a query for each section of the outlines. For each query, we retrieved and ranked the relevant paragraphs using different IR models. Finally, we filled the outlines using these relevant paragraphs according to three different strategies.

[3] `http://data.allenai.org/tqa/`, last access on October 22, 2019.

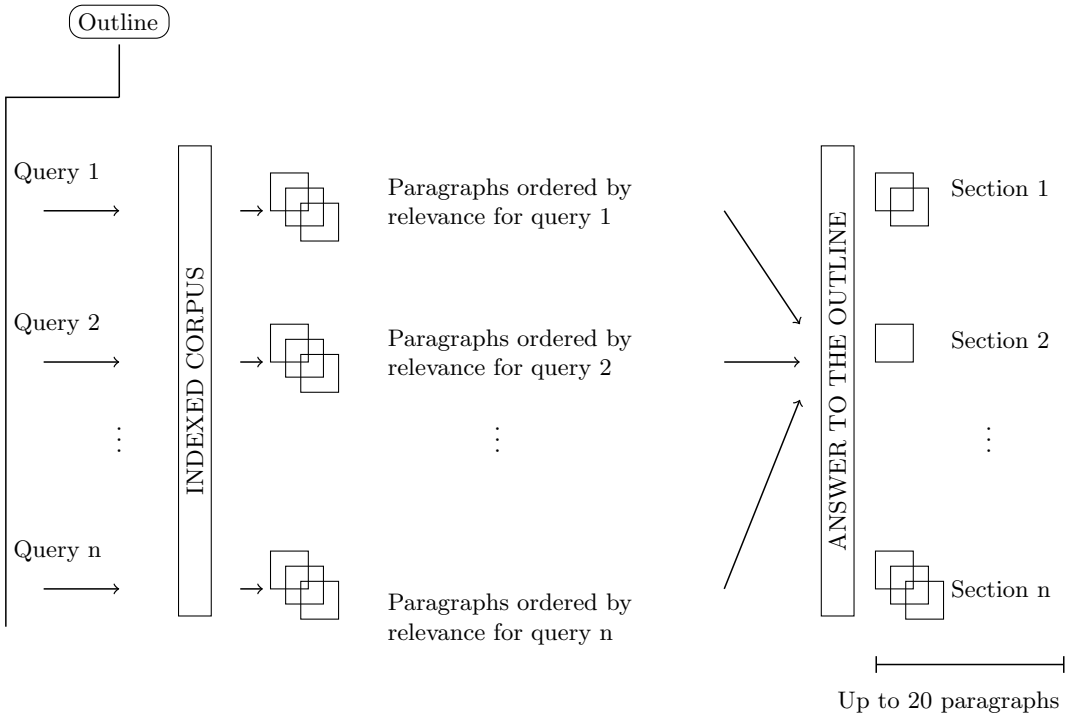

Fig. 2. General approach used by the IRIS team for the TREC CAR track

3.2.1 Indexing

The TREC CAR 2019 track provided about 30 million paragraphs from Wikipedia. We indexed these paragraphs using the Terrier retrieval system [14]. We used the default Terrier indexing settings.

3.2.2 Retrieving relevant paragraphs

For each outline, the different sections to be answered are provided by the organizers. We built our queries from the outline and its different sections. For each section, we concatenated its title with the title of the corresponding outline in order to obtain the queries.

At this point, we thus have the index of the corpus of paragraphs and our different queries for which we want to retrieve the most relevant paragraphs.

In order to retrieve the most relevant paragraphs, we used the models already implemented in Terrier : BB2, BM25, DFR_BM25, DLH, DLH13, DPH, DFRee, Hiemstra_LM, IFB2, In_expB2, In_expC2, InL2, LemurTF_IDF, LGD, PL2, and TF_IDF. Each of these models returned a ranking of relevant paragraphs to each query. We then used the CombMNZ function to keep only one ranking from all models. The CombMNZ function returns a ranking by combining the rankings from the different models [15]:

$$ScoreCombMNZ_s(p) = (\sum_{m=1}^{n} Score_{ms}(p)) \cdot Count_s(p) \quad (1)$$

where n is the number of weighting models, s is the concerned section, $Score_{ms}(p)$ is the score calculated by the model m for the paragraph p for the section s, and $Count_s(p)$ is the number of models that have retrieved the paragraph p for the section s.

As a result, we obtained a ranking of paragraphs for each query, i.e., for each section of an outline.

3.2.3 Answer to the outline

Following the guidelines, we answered each outline using a maximum of 20 paragraphs. To choose these 20 paragraphs from the different rankings computed for all the sections, we applied three different strategies:

- The first strategy uses a classic Round-Robin algorithm. The best paragraph of each section is selected in an iterative way until the maximum of 20 paragraphs is reached with the constraint that one paragraph can only appear once in the outline. Figure 3 is an exemplification of this strategy. For the outline "Word climates", the different sections of the outline are depicted. For each section, the ranking is presented with the rank, the paragraph ID, and the CombMNZ score of the paragraph for the query corresponding to the section. Highlighted lines represent the paragraphs kept to answer the outline. This strategy has some limits, notably regarding the ranking across sections: one paragraph may have a higher score for section A than another one for section B but as it is less well ranked it will not be kept for filling the outline. To illustrate this limit, in Figure 3, the paragraph Para_4 of the "Tropical climates" section has a relevance score of 550 and is kept in the final result while the paragraph Para_18 of the "Continental climates" section has a relevance score of 950 and is not kept.
- To overcome this problem, we took into account in strategies 2 and 3 the ranking but also the CombMNZ score of the paragraphs and the number of paragraphs of the section already recovered to answer the outline. For each section, a candidate paragraph is chosen (the one among the highest unrecovered paragraphs in the ranking for the section). For all the candidates (one per section), a function is applied, and the one that maximizes the score of the function is selected. This operation is repeated until the limit of 20 paragraphs is reached.

  For our second strategy, we used the following formula:

$$Function(candidate_s) = se_s(candidate_s) - ce_s(candidate_s) - nb_s \quad (2)$$

with $candidate_s$ the candidate paragraph for section s, se_s the CombMNZ score of the paragraph for the section scaled between 0 and 1, ce_s the ranking of the paragraph for the section scaled between 0 and 1, and nb_s the number of already retrieved paragraphs for the section, scaled between 0 and 1.

To scale between 0 and 1, the MinMaxScaler function from sklearn[4] was applied for all the selected candidates:

$$scale(x) = \frac{x - min(X)}{max(X) - min(X)} \quad (3)$$

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`, last access on October 22, 2019.

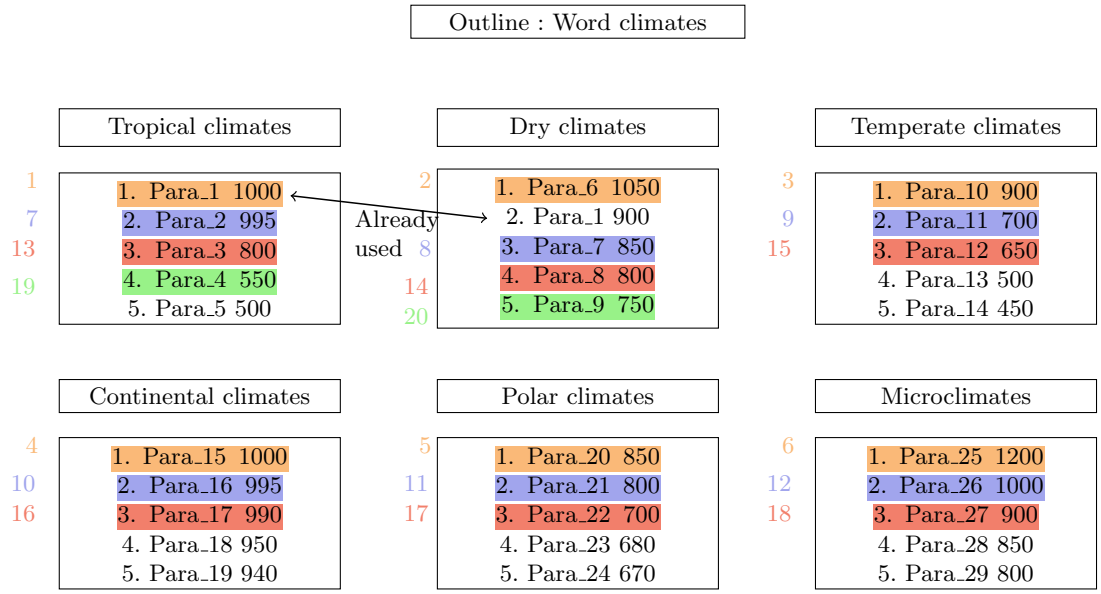

Fig. 3. Illustration of the algorithm used to return paragraphs according to the outline and sections.

with X a vector, x a value of X, $min(X)$ the minimum value of X and $max(X)$ the maximum value of X.

At last, for our third strategy, we used the following formula:

$$Function(candidate_s) = ln(ScoreCombMNZ_s(candidate_s)) - c_s(candidate_s) - nb_s \quad (4)$$

with $ScoreCombMNZ_s$ the paragraph score for section s, c_s the ranking of the paragraph for section s and nbr_s the number of paragraphs in section s already retrieved.

3.3 Runs

We submitted three runs:

- Our run using the Round-Robin strategy is named `IRIT_run1`.
- Our run that uses strategy 2 is named `IRIT_run2`.
- Finally, our run that uses strategy 3 is named `IRIT_run3`.

Figure 4 shows the results of the track for 2019. These results do not consider the passage ordering task but the section-level ranking performance. This explains the similar scores for the three runs. In other words, only the approaches described up to section 3.2.3 are evaluated in the task.

4 Conclusion

In this paper we presented our approach for the TREC IS 2019 task, which aims at categorizing tweets from streams in order to help emergency services in case of incidents.

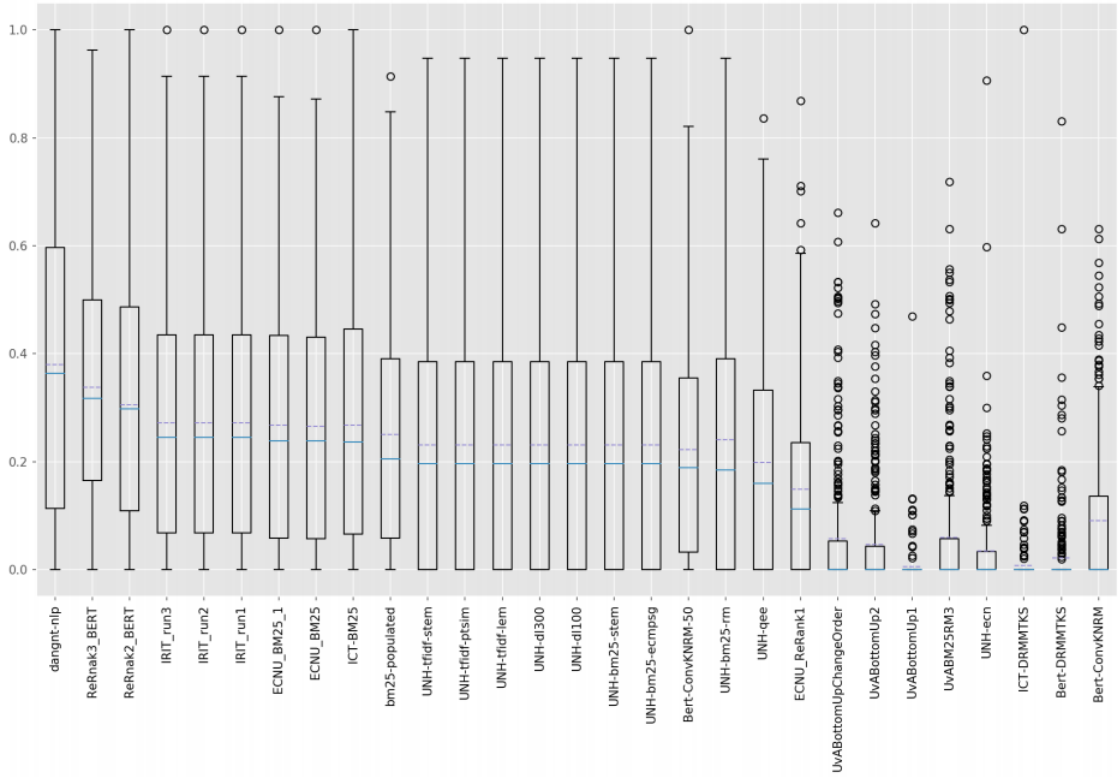

Fig. 4. Section-level ranking performance in NDCG, TREC CAR 2019

We also presented our approach for the TREC CAR 2019 task, which aims to answer complex questions.

References

1. Carlos Castillo. *Big Crisis Data*. Cambridge University Press, 2016.
2. Richard Mccreadie, Cody Buntain, and Ian Soboroff. Trec incident streams: Finding actionable information on social media, March 2019.
3. Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly, 2009.
4. Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.
5. Abu Nowshed Chy, Umme Aymun Siddiqua, and Masaki Aono. Neural networks and support vector machine based approach for classifying tweets by information types at TREC 2018 incident streams task. In Ellen M. Voorhees and Angela Ellis, editors, *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*, volume Special Publication 500-331. National Institute of Standards and Technology (NIST), 2018.

6. Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer, editors, *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 363–370. The Association for Computer Linguistics, 2005.
7. Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In Marti A. Hearst and Mari Ostendorf, editors, *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics, 2003.
8. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
9. Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232, 10 2001.
10. Jonathan L. Lustgarten, Vanathi Gopalakrishnan, Himanshu Grover, and Shyam Visweswaran. Improving classification performance with discretization on biomedical datasets. In *AMIA 2008, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 8-12, 2008*. AMIA, 2008.
11. Alisa A. Vorobeva. Influence of features discretization on accuracy of random forest classifier for web user identification. In *20th Conference of Open Innovations Association, FRUCT 2017, St. Petersburg, Russia, April 3-7, 2017*, pages 498–504. IEEE, 2017.
12. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.
13. Aniruddha Kembhavi, Min Joon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5376–5384. IEEE Computer Society, 2017.
14. Craig Macdonald, Richard McCreadie, Rodrygo LT Santos, and Iadh Ounis. From puppy to maturity: Experiences in developing terrier. *Proc. of OSIR at SIGIR*, pages 60–63, 2012.
15. Joseph A. Shaw and Edward A. Fox. Combination of multiple searches. In Donna K. Harman, editor, *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume Special Publication 500-225, pages 105–108. National Institute of Standards and Technology (NIST), 1994.