Yandex at TREC 2011 Microblog track

Zlata Obukhovskaya, Konstantin Pervyshev
Andrey Styskin, Pavel Serdyukov
Yandex

1 Introduction

Building microblog search is a challenging task in many ways. One of its most exciting aspects is the creation of reliable search engine architecture. However in our work we focused on tweet retrieval, utilizing relevance signals coming from various sources related to a tweet and learning to rank.

For ranking-related tasks we decided to rely on the following data sources:

- Text sources: tweet itself and the title of the linked URL
- Crowd activity (retweets)

We assumed that the number of retweets should be a good indicator of social interest to the particular document and the retweet prediction would be a useful feature for ranking. Also we encountered a problem with low recall of TF-based retrieval on small-sized twitter documents. This prompted us to use flexible term weighting models and query expansion to avoid missing relevant results.

2 Search Framework

For our experiments we developed an offline search system that operated on MapReduce cluster. On the way we faced various problems related to inverted indexing and search.

First, we had to calculate query-specific, time-specific and other statistics without re-indexing the entire data each time. So we tried to speed up the whole cycle of search and simplify the process of adding new features. As a result our search framework was able to calculate feature values for query-tweet pairs by a full scan of twitter corpus, aggregate them, if necessary, and re-use during any consequent full scan passes. So the search process represented a series of full scan and aggregation phases. All test topics were processed in parallel what took only 4 minutes on a cluster of 500 machines.

3 Features

In this section we describe features and the details of ranking model. Experiments results are discussed in Section 4, Section 5.

3.1 Document-Query Features and Query Expansion

TF-based textual features do not work well for twitter statuses since tweets are much shorter than regular web documents. In order to get as much signal from textual features as possible we had to use query expansion.

Our query expansion technique is based on pseudo-relevance feedback[1]. For each word in the results retrieved in the first search pass the relevance score is calculated. Relevance score is a probability of word being sampled from the same distribution as the words of query. Words are sorted by these scores in descending order. Then every word is added to the expansion while the ratio of the next word's score and the current word's score is less than some threshold. Which was empirically chosen to be 0.3.

Some of our textual features use expansions.

#	ID	Description
0	SumIDF_Q	Sum of query words IDFs
1	SumIDF_QEx	Sum of expanded query words IDFs
2	3Gram_Q	Jaccard index for the sets of character-level trigrams of document and query
3	3Gram_QEx	Jaccard index for the sets of character-level trigrams of document and expanded query
4	NWords_Q	Number of words in query
5	NWords_QEx	Number of words in expanded query
6	SumPRel_QEx	Sum of relevance scores for the words added by query expansion

Table 1: Textual Features

3.2 Static Features

We used some external information for creating two groups of query-independent features. Most of the query-independent features mentioned below were implemented in related work[2] (marked with stars).

#	ID	Description
7	TweeLen*	Tweet length in characters
8	NExcl*	Number of question and exclamation marks in the tweet
9	NSmiles*	Number of smiles in the tweet
10	NOutLiks*	Number of outgoing links in the tweet
11	NHTags*	Number of hashtags

12	AvgWordLen*	Average length of word in characters
13	SelfCentrism*	Number of first-person pronouns (like: I, me, mine, my)
14	PosSent*	Number positive words in document
15	NegSent*	Number of negative words in document
16	NStopword*	Number of stop words
17	TextDiversity	Ratio of the number of unique word-level trigrams to total number of words in the tweet

Table 2: Informativness features

#	ID	Description
18	NFollowers*	Number of user's followers
19	NTweets*	Number of user's tweets
20	NFollowees*	Number of user's followees
21	NListed*	Number of groups where user was added
22	Verified*	True if users's account is verified
23	FollowersRatio	Followees to followers ratio
24	NLastStatRT	Number of the retweets of the most recent user's status

Table 3: Social features

4 Ranking Function

In this section we describe how we trained ranking functions for TREC runs. We analyze improvements that we made after NIST assessments were distributed. Also we discuss the influence that our interestingness prediction (see Section 4.2) have over ranking quality.

We used Gradient Boosted Decision Trees (GBDT) tool for training. Query expansion was used in all our runs.

The learning pool that we used was formed as follows. For each of the 12 example topics provided by NIST we selected about 80 tweets that had at least one of the text feature values greater than a certain threshold. All those tweets were assessed as either “relevant” or “not relevant”.

4.1 Run 1 and Run 2

For both runs we trained regression on textual features (see Section 3.1). The first and the second attempt differed in the number of the iterations made by the learning algorithm. Results are in Table 5.

4.2 Run 3 and Run 4

We used regression for run 3 and classification for run 4. In addition to textual features (see Section 3.1) we built extra features based on retweet prediction.

Our intuitions about using retweets for interestingness prediction are given in Section 5.1.1.

For creating retweet predictors we trained several classifiers with retweet classes as targets (0 for 0 retweets, 1 for 0-10 retweets, 2 for >10 retweets) and combining various features sets and learning samples.

#	ID	Features set	Account type	Followers
25	Inf_nVer_L200	informativeness	non-verified	<500
26	Inf_Ver_M200	informativeness	verified	>500
27	Soc_nVer_L500	social	non-verified	<500
28	Soc_Ver_M500	social	verified	>500

Table 4: Retweet predictors

4.3 TREC results

Our results, according to NIST.

	run 1	run 2	run 3	run 4
`P_30 (relevant)`	0.2027	0.2156	0.2190	0.2388
`P_30 (highly relevant)`	0.0667	0.0697	0.0515	0.0646

Table 5: TREC runs results

4.4 Quality Improvement

Having received the 50 topic tweet assessment provided by NIST we were able to improve our ranking function. Using positive examples from the assessment we increased `P_30` from 0.2388 to 0.2923 for relevant tweets and from 0.0697 to 0.1011 for highly relevant ones. Using both positive and negative examples we managed raise `P_30` even more.

Here, we discuss ranking function that was learned on random negative examples. This formula was trained as follows. From the 16 million tweet collection, we chose those tweets that contained at least one word from a topic extension. Of those tweets all relevant tweets (1800 tweets) and some randomly chosen were selected for our training pool.

For the resulting formula a 5-fold cross-validation evaluation on 50 topics was conducted. As can be seen, `P_30` strongly depends on the number of negative examples (i.e. non-relevant tweets) in the training pool:

negative examples	50%	80%	94%	98.5%
`P_30 (relevant)`	0.1998	0.2462	0.2827	0.2923
`P_30 (highly relevant)`	0.0697	0.0838	0.0919	0.1011

The best `P_30` value is obtained when negative examples amounts to at least 98.5% of all examples.

Our new ranking function has P_30 that is noticeably higher than before. The main reason for that is that our official runs were trained with much fewer negative examples (76%) than is optimal. Another explanation is that the new ranking function was trained with 40 topics, while our runs were trained with 12 topics only.

We conducted experiments to evaluate the dependence of P_30 on the number of topics in the pool. For 12 topics, the resulting P_30 equals to 0.2684 and noticeably depends on the particular topics of which the pool is formed.

	P_10	P_30	P_50	P_100
run 1	0.3653	0.3122	0.2914	0.2308
run 2	0.4429	0.3469	0.2931	0.2276
run 3	0.2531	0.2252	0.2082	0.1749
run 4	0.2939	0.2483	0.2196	0.1796
	NDCG_10	NDCG_30	NDCG_50	NDCG_100
run 1	0.3806	0.4004	0.4277	0.4625
run 2	0.4763	0.4529	0.4578	0.4865
run 3	0.2554	0.2860	0.3137	0.3530
run 4	0.3067	0.3228	0.3416	0.3753

Table 6: Quality of ranking functions trained using NIST assessments

5 Further Experiments

5.1 Experiments outline

For further experiments we reworked our search engine retrieval algorithm. In this section we ran experiments using simple search method and search with query and outer links expansions. Combinig features described in Section 3, and Section 5.1.1 we made a number of ranking functions using regression with GBDT tool.

The following sections describe details of the experiments and quality evaluation.

5.1.1 Artificially Created Features or Interestingness Predictors

The original lack of labels lead us to the idea of mining it from twitter statuses. The retweets seemed a good crowdsourced indicator of tweets interestingness. So we used retweets as targets while training retweet prediction funtion on the various sets of features (see Section 3). We called these artificially created features "interestingness predictors". The outcome of the predictors become new features values for training final ranking functions.

The natural consideration about using retweets as labels was that its interestingness signal may somehow be noisy. Shouldn't popular users be retweeted only because of popularity?

We tried to clear the signal by training predictors on the special tweet samples. Samples were created by splitting the whole corpus into parts relying on users social features.

#	ID	Features set	Account type	Followers
29	InfSoc_nVer_M200	informativeness, social	non-verified	>200
30	InfSoc_nVer_L200	informativeness, social	non-verified	<200
31	InfSoc_nVer_M500	informativeness, social	non-verified	>500
32	InfSoc_nVer_L500	informativeness, social	non-verified	<500
33	InfSoc_nVer_M1000	informativeness, social	non-verified	>1000
34	InfSoc_nVer_L1000	informativeness, social	non-verified	<1000
35	InfSoc_Ver	informativeness, social	verified	
36	Inf_nVer_M200	informativeness	non-verified	>200
37	Inf_nVer_L200	informativeness	non-verified	<200
38	Inf_nVer_M500	informativeness	non-verified	>500
39	Inf_nVer_L500	informativeness	non-verified	<500
40	Inf_nVer_M1000	informativeness	non-verified	>1000
41	Inf_nVer_L1000	informativeness	non-verified	<1000
42	Inf_Ver	informativeness	verified	
43	Soc_nVer_M200	social	non-verified	>200
44	Soc_nVer_L200	social	non-verified	<200
45	Soc_nVer_M500	social	non-verified	>500
46	Soc_nVer_L500	social	non-verified	<500
47	Soc_nVer_M1000	social	non-verified	>1000
48	Soc_nVer_L1000	social	non-verified	<1000
49	Soc_Ver	social	verified	

Table 7: More interestingness predictors

5.1.2 Quality Evaluation

We evaluated MAP, P_30 and BPREF measures with TREC evaluation tool using 5-fold cross-validation on 50 topics. All quality numbers in the following sections are averaged over folds.

5.2 Simple Search

5.2.1 Textual Features

In this experiment we learned ranking function with textual (see Table 1) features that do not use expansions.

Features set	P_30	MAP	BPREF
0,2,4	0.4103	0.4112	0.1804
0	0.3677	0.3616	0.1006
2	0.1646	0.1503	0.0058
4	0.3044	0.3104	0.0763

Table 8: No expansions, textual features

5.2.2 Informativeness & Social Features

This experiment was generally meant to measure static features (see Section 3.2) strength.

Features set	P_30	MAP	BPREF
7	0.1092	0.1190	0.0007
8	0.0333	0.0748	0.0007
9	0.0329	0.0818	0.0007
10	0.0333	0.0822	0.0007
11	0.2870	0.2656	0.0679
12	0.0639	0.0966	0.0014
13	0.0341	0.0793	0.0007
14	0.0192	0.0870	0.0003
15	0.0239	0.0675	0.0007
16	0.0412	0.0842	0.0007
17	0.0979	0.1240	0.0031
18	0.0364	0.0752	0.0010
19	0.0356	0.0740	0.0006
20	0.0247	0.0693	0.0007
21	0.0655	0.0895	0.0047
22	0.0652	0.0898	0.0011
23	0.0820	0.0950	0.0031
24	0.0382	0.0823	0.0007

Table 9: No expansions, informativeness & social features strength

5.3 Query & Outer Link Expansion

For the following set of experiments we used search with pseudo-relevance feedback query expansion (see Section 3.1). Also we fetched titles of 20% of documents linked from tweets and made pseudo tweets out of them.

5.3.1 Textual Features

Considering strength of features (compare Table 10 to Table 8) and number of relevant documents in search results (see Table 11) it seems that query expansions do not help to find more relevant documents, though features based on

Features set	P_30	MAP	BPREF
0-6	0.4744	0.4810	0.2567
0	0.3646	0.3607	0.1052
1	0.3809	0.3559	0.0485
2	0.1773	0.1742	0.0105
3	0.1467	0.1465	0.0051
4	0.2761	0.2675	0.0809
5	0.2571	0.2244	0.0109
6	0.2344	0.2057	0.0066

Table 10: Expansions, textual features

Number of search results	50	100	500	1000
`expansion`	1005	1563	2421	2497
`no expansion`	894	1397	2405	2497

Table 11: Number of relevant documents in the search results, summarized over 50 topics

expansions (1, 3, 5, 6) help to rank search results.

5.3.2 Informativeness & Social Features

This experiment shows how relatively weak features (see Table 9) help to improve quality.

Features set	P_30	MAP	BPREF
0-6	0.4744	0.4810	0.2567
0-6,7	0.5010	0.5069	0.2698
0-6,8	0.4854	0.4906	0.2622
0-6,9	0.4738	0.4821	0.2666
0-6,10	0.4959	0.5025	0.3228
0-6,11	0.4864	0.4876	0.2491
0-6,12	0.5104	0.5171	0.3136
0-6,13	0.4830	0.4858	0.2630
0-6,14	0.4843	0.4914	0.2717
0-6,15	0.4861	0.4899	0.2702
0-6,16	0.5024	0.5035	0.2685
0-6,17	0.4806	0.4852	0.2566
0-6,18	0.5024	0.5026	0.2732
0-6,19	0.5011	0.5005	0.2815
0-6,20	0.4981	0.5003	0.2707
0-6,21	0.5001	0.5022	0.2836
0-6,22	0.4742	0.4826	0.2683
0-6,23	0.4975	0.4990	0.2844
0-6,24	0.4810	0.4848	0.2551

Table 12: Expansions, informativeness & social features strength

5.3.3 Predictor Features

Predictors in Table 13 were trained as the regression to predict number of retweets. But none of them gave quality improvement.

Features set	P_30	MAP	BPREF
0-6	0.4744	0.4810	0.2567
0-6,29-49	0.4711	0.4784	0.2545
0-6,29	0.4648	0.4459	0.2023
0-6,30	0.4575	0.4599	0.2346
0-6,31	0.4596	0.4674	0.2600
0-6,32	0.4684	0.4661	0.2403
0-6,33	0.4618	0.4669	0.2659
0-6,34	0.4616	0.4580	0.2375
0-6,35	0.4646	0.4655	0.2620
0-6,36	0.4535	0.4509	0.2111
0-6,37	0.4635	0.4630	0.2396
0-6,38	0.4557	0.4445	0.2213
0-6,39	0.4599	0.4642	0.2461
0-6,40	0.4451	0.4322	0.2061
0-6,41	0.4530	0.4500	0.2224
0-6,42	0.4518	0.4450	0.2166
0-6,43	0.4647	0.4590	0.2456
0-6,44	0.4601	0.4607	0.2537
0-6,45	0.4604	0.4635	0.2606
0-6,46	0.4625	0.4627	0.2360
0-6,47	0.4602	0.4562	0.2590
0-6,48	0.4574	0.4558	0.2466
0-6,49	0.4551	0.4576	0.2324

Table 13: Expansions, predictors strength

6 Conclusions

In this paper we shared our experience of creating microblog search.

Query expansions didn't help to improve search recall, but textual features based on expansions improved ranking. Query-independent features based on document text itself and its author's profile gave desired quality increase. Using retweets for predicting interestingness also didn't help in our struggle for better ranking and retrieval. Though we inclined to believe that further research on correlation between various values mined from twitter (for example, correlation between number of followers and retweets) will give new insights on building microblog search.

References

[1] W. B. Croft, V. Larvrenko. *Relevance-Based Language Models.* Center for Intelligent Information Retrieval, Department of Comupter Science, University of Massachusetts, Amherst, 2001.

[2] C. Castillo, M. Mendoza, B. Poblete. *Information Credibility on Twitter.* WWW 2011 Session: Information Credibility.

[3] M. Efron and G. Golovchansky. *Estimation Methods for Ranking Recent Information.* In SIGIR, 2011.

[4] N. Naveed, T. Gottron, J. Kunegis and A. Alhadi. *Searching microblogs: coping with sparsity and document quality.* in 20th ACM Conference on Information and Knowledge Management (CIKM 2011).