A Syntax-Aware Re-ranker for Microblog Retrieval

Aliaksei Severyn
DISI, University of Trento
severyn@disi.unitn.it

Alessandro Moschitti
QCRI, Qatar
amoschitti@qf.org.qa

Manos Tsagkias
University of Amsterdam
e.tsagkias@uva.nl

Richard Berendsen
University of Amsterdam
r.w.berendsen@uva.nl

Maarten de Rijke
University of Amsterdam
derijke@uva.nl

ABSTRACT

We tackle the problem of improving microblog retrieval algorithms by proposing a robust structural representation of (query, tweet) pairs. We employ these structures in a principled kernel learning framework that automatically extracts and learns highly discriminative features. We test the generalization power of our approach on the TREC Microblog 2011 and 2012 tasks. We find that relational syntactic features generated by structural kernels are effective for learning to rank (L2R) and can easily be combined with those of other existing systems to boost their accuracy. In particular, the results show that our L2R approach improves on almost all the participating systems at TREC, only using their raw scores as a single feature. Our method yields an average increase of 5% in retrieval effectiveness and 7 positions in system ranks.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.3 Information Search and Retrieval

Keywords

Microblog search; semantic modeling; re-ranking

1. INTRODUCTION

Social media has become part of our daily lives, and is increasingly growing into the main outlet for answering various information needs, e.g., the query, *Facebook privacy*, may be answered by the following tweet: *Facebook Must Explain Privacy Practices to Congress http://sns.ly/2Qbry7*. Such queries have proven difficult to answer with a single retrieval model, and lead to models that learn to combine a large number of rankers. Learning to rank (L2R) methods have been shown to improve retrieval effectiveness and they have recently been used for ranking short documents from social media. However, L2R suffers from an important drawback: different training data is needed for different applications. The required amount of training data critically depends on the task being tackled and the *quality* of the used text representations, e.g., lexical features are less powerful than search engine scores or other meta-features. Optimal representations require considerable effort to be designed and implemented. Hence, flexible and adaptable features can be valuable for rapid and effective designs of L2R systems.

Previous work has shown that one source of more *adaptable* features comes from structural relations between object pairs [6], which in the case of text mainly refers to its syntactic structure. Unfortunately, the latter is subject to errors when it is automatically generated. This problem is exacerbated when we deal with informal and unedited text typically prominent in social media. Most importantly, it is not clear which part of the structure should be considered to design effective features.

We tackle the problems noted above in the context of recent TREC Microblog retrieval tasks by proposing relational shallow syntactic structures to represent (query, tweet) pairs. Instead of trying to explicitly encode salient features from syntactic structures, we opt for a structural kernel learning framework, where the learning algorithm operates in rich feature spaces of tree fragments automatically generated by expressive tree kernel functions.

The following characterizes our approach: (i) it uses shallow syntactic parsers developed for social media, which are robust and shown to be accurate in such domains; (ii) tree kernels implicitly generate all possible tree fragments, thus all of them are used as features by the learning algorithm, solving the problem of engineering task-specific features. We design experiments using the 2011 and 2012 editions of the TREC Microblog track to verify the following: (i) relational syntactic features produced by a shallow syntactic parser are effective for L2R; (ii) our automatic feature engineering approach based on structural kernels is accurate and produces general features, which are complementary to those typically used in L2R models; and (iii) our structural representations can easily be combined with existing systems to boost their accuracy.

Our results show that employing relational syntactic structures improves on almost all the participating systems by only using their raw scores along with our L2R model based on relational syntactic structures. Our method boosts retrieval effectiveness by more than 5% on average and improves the rankings of participating systems by at least 7 positions on average.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '14, July 06–11, 2014, Gold Coast, QLD, Australia.
Copyright 2014 ACM 978-1-4503-2257-7/14/07 ... $15.00.
http://dx.doi.org/10.1145/2600428.2609511 .

2. A SYNTAX-AWARE RE-RANKER

Our syntax-aware re-ranker consists of two components: (i) a syntactic model that encodes tweets into shallow linguistic trees to ease feature extraction, and (ii) a tree kernel learning framework that computes similarities between (query, tweet) pairs. We also define a shallow tree kernel to enable efficient kernel computations.

2.1 A syntactic model for tweets

Our approach to extract features from (query, tweet) pairs goes beyond traditional feature vectors. We employ structural syntactic models (`STRUCT`) that encode each tweet into shallow syntactic

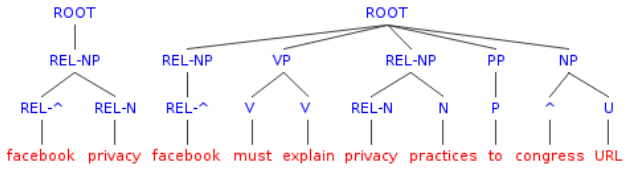

Figure 1: Shallow tree representation for an example (query, tweet) pair: (*"Facebook privacy", "Facebook Must Explain Privacy Practices to Congress http://sns.ly/2Qbry7"*) (in the original tree we use word stems). Part-of-speech tag ^ refers to a common noun. Note that an additional `REL` tag links the words (stems) common between the query and the candidate tweet, *Facebook* and *privacy*).

trees. The latter are input to tree kernel functions for generating structural features. Our structures are specifically adapted to the noisy tweets and encode important query/tweet relations.

In particular, our shallow tree structure (inspired by [6–8]) is a two-level syntactic hierarchy built from word lemmas (leaves) and part-of-speech tags that are grouped into chunks (Fig. 1). While full syntactic parsers would significantly degrade in performance on noisy texts such as tweets, our choice for shallow structure relies on simpler and more robust components: a part-of-speech (POS) tagger and a chunker. For POS tagging we use the CMU tagger [3] trained on Twitter data and an off-the-shelf OpenNLP chunker.

Fig. 1 provides an example of a candidate (query, tweet) pair each of which is encoded into a shallow linguistic structure. To upweight the tree fragments spanning words that are found in both the query and the tweet we introduce a special `REL` tag at the level of part-of-speech and chunk nodes. This step is important to generate syntactic patterns that carry additional semantics of sharing common terms between a query and a tweet. To find matching word pairs we lowercase and stem words and use plain string matching.

2.2 Learning

We employ a pointwise approach to re-ranking where a binary classifier is used to learn a model to discriminate between relevant and non-relevant (query, tweet) pairs. The prediction scores from a classifier are then used to re-rank candidates. We define a novel and efficient tree kernel function, namely, **Sh**allow syntactic **T**ree **K**ernel (SHTK), which is as expressive as Partial Tree Kernel (PTK) [4] to handle feature engineering over the structural representations of the `STRUCT` model. For feature vectors we use a linear kernel.

Computing similarity between (query, tweet) pairs. A typical kernel machine classifies a test input example $\mathbf{x}$ using the following prediction function: $h(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$, where α_i are the model parameters estimated from the training data, y_i are target variables, $\mathbf{x}_i$ are support vectors, and $K(\cdot,\cdot)$ is a kernel function that computes the *similarity* between two input objects.

We represent each (query, tweet) pair $\mathbf{x}$ as a triple composed of a query tree $\mathbf{T}_q$ and a tweet tree $\mathbf{T}_{tw}$ together with a traditional feature vector $\mathbf{v}$, i.e., $\mathbf{x} = \langle \mathbf{T}_q, \mathbf{T}_{tw}, \mathbf{v} \rangle$. Given two (query, tweet) pairs $\mathbf{x}^i$ and $\mathbf{x}^j$, we define the following similarity kernel:

$$\begin{aligned} K(\mathbf{x}^i, \mathbf{x}^j) \quad = \quad & K_{TK}(\mathbf{T}^i_q, \mathbf{T}^j_q) + K_{TK}(\mathbf{T}^i_q, \mathbf{T}^j_{tw}) \\ & + K_{TK}(\mathbf{T}^i_{tw}, \mathbf{T}^j_{tw}) + K_{TK}(\mathbf{T}^j_q, \mathbf{T}^i_{tw}) \\ & + K_{\mathrm{v}}(\mathbf{v}^i, \mathbf{v}^j), \end{aligned} \quad (1)$$

where K_{TK} computes a tree kernel similarity between linguistic trees and K_{v} is a kernel over feature vectors. It computes an all-vs-all tree kernel similarity between two (query, tweet) pairs.

Shallow syntactic tree kernel. Following the convolution kernel framework, we define the new SHTK function from Eq. 1 to compute the similarity between tree structures. It counts the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. The general equation for Convolution Tree Kernels is: $K_{TK}(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the nodes in T_1 and T_2, respectively, and $\Delta(n_1, n_2)$ is equal to the number of common fragments rooted in the n_1 and n_2 nodes, according to several possible definitions of the atomic fragments.

To speed up the computation of K_{TK}, we consider pairs of nodes (n_1, n_2) belonging to the same tree level. Thus, given H, the height of the `STRUCT` trees, where each level h contains nodes of the same type, i.e., chunk, POS, and lexical nodes, we define SHTK as the following:

$$K_{SHTK}(T_1, T_2) = \sum_{h=1}^{H} \sum_{n_1 \in N^h_{T_1}} \sum_{n_2 \in N^h_{T_2}} \Delta(n_1, n_2), \quad (2)$$

where $N^h_{T_1}$ and $N^h_{T_2}$ are sets of nodes at height h.

The above equation can be applied with any Δ function. To have a more general and expressive kernel, we use Δ from PTK, which employs subsequence kernels, thus making it possible to generate child subsets of the two nodes, i.e., it also allows for gaps, which makes matching of syntactic patterns less rigid.

The resulting SHTK is a special case of PTK [4], adapted to the shallow structural representation `STRUCT`. When applied to `STRUCT` trees, SHTK computes the same feature space as PTK, but faster (on average). Unlike PTK, where all combinations of node pairs are considered, the kernel definition in (2) constrains the node pairs being considered to be from the same level, i.e., the matched nodes have to be of the same type—chunk, POS or lexicals. Hence, the number of node pairs considered for matching by SHTK is smaller, which results in faster kernel evaluation.

Finally, given its recursive definition in Eq. 2 and the use of subsequences (with gaps), SHTK can derive useful dependencies between its elements. E.g., it will generate the following subtree fragments (in nested parenthesis format):

1. `[ROOT [REL-NP[REL-^ facebook]][VP V][REL-NP [REL-N privacy]]]`
2. `[ROOT [REL-NP[REL-^]][VP V][REL-NP [REL-N]]]`
3. `[ROOT [REL-NP][VP][REL-NP]]`
4. `[ROOT [VP[V explain]][NP[N privacy]]].`

Subtree 3 generalizes Subtree 2, which, in turn, generalizes Subtree 1. These structures are interesting when paired with the query structure. E.g., given the query pattern, `[REL-NP [REL-^][REL-N]]`, which means a famous proper noun (^) followed by a noun, if the tweet contains Subtree 2, i.e., a famous proper noun (matched in the query) followed by a verbal phrase (VP) and a common noun (also matched in the query), the candidate tweet may be relevant.

3. EXPERIMENTS AND EVALUATION

To evaluate the utility of our structural syntactic re-ranker for microblog search we focus on the 2011 and 2012 editions of the ad-hoc retrieval task at TREC microblog tracks [5, 9]. Our main research question is: Does the use of relational syntactic features produced by our shallow syntactic parser, and the automatic feature engineering approach based on structural kernels lead to improvements in state-of-the-art L2R and retrieval algorithms?

To answer this question, we test our model in two settings. In the first, we re-implement an accurate recent L2R-based approach and add our features alongside its features. This will allow us to see directly if our features are complementary to the other features. We

opted for the L2R approach in [2] ("the UvA model"), because of its comprehensiveness. It uses pseudo-test collections [1] to learn to fuse ten well-established retrieval algorithms and implements a number of query, tweet, and query-tweet features. It is a strong baseline, its performance ranks sixth and 26th in the 2011 and 2012 editions of the microblog track, respectively. In the second setting, we use the participant systems in the TREC microblog task as a black-box, and implement our model on top of them using only using their raw scores (ranks) as a single feature in our model. This allows us to see whether our features add information to the approaches these retrieval algorithms use.

3.1 Experimental setup

Dataset. Our dataset is the tweet corpus used in the TREC Microblog track in both 2011 (TMB2011) and 2012 (TMB2012). It consists of 16M tweets spread over two weeks, and a set of 49 (TMB2011) and 60 (TMB2012) timestamped topics. We minimally preprocess the tweets—we normalize elongations (e.g., sooo $\rightarrow$ so) and normalize URLs and author ids. For the second set of experiments, we also use the system runs submitted at TMB2011 and TMB2012, which contain 184 and 120 models, respectively.

Training and testing an L2R algorithm. For learning to rank we use SVM-light-TK[1] with no parameter tuning. In our first set of experiments, we train on TMB2011 topics, test on TMB2012 topics, and vice versa. In the second set, where we build upon the TREC participant runs, we train our system only on the runs submitted at TMB2011, and test on the TMB2012 runs. We focus on one direction only to avoid training bias, since TMB2011 topics were already used for learning systems in TMB2012.

Feature normalization. When combining our features with those of the UvA model, while training and testing we use the features of the latter model as $\mathbf{v}$ in Eq. 1; these features are already normalized. In contrast, we use the output of participant systems as follows. We use rank positions of each tweet rather than raw scores, since scores for each system are scaled differently, while ranks are uniform across systems. We apply the following transformation of the rank r: $1/\log(r+1)$. In the training phase, we take the top {10, 20, or 30} systems from the TMB2011 track (in terms of P@30). For each (query, tweet) pair we average the transformed rank over the top systems to yield a single score. This score is then used as a single feature in $\mathbf{v}$ from Eq. 1. In the testing phase, for each participant system we want to improve, we use the transformed rank of the (query, tweet) pairs as the single feature in $\mathbf{v}$.

Evaluation. We report on the official evaluation metric for the TREC 2012 Microblog track, i.e., precision at 30 (P@30), and also on mean average precision (MAP). Following [2, 5], we regard minimally and highly relevant documents as relevant and use the TMB2012 evaluation script. For significance testing, we use a pairwise t-test, where $\triangle$ and ▲ denote significance at $\alpha = 0.05$ and $\alpha = 0.01$, respectively. Triangles point up for improvement over the baseline, and down otherwise. We also report the improvement in the absolute rank (R) in the official TMB2012 ranking.

3.2 Results

Table 1 lists the outcome of our first set of experiments, where we use our syntactic features alongside the features of the UvA model. It shows the obtained MAP and P@30 scores when we train on TMB2011 and test on TMB2012 topics, and vice versa. The STRUCT model yields a significant improvement in P@30 and MAP scores on TMB2012 pushing up the system by 15 positions in the official ranking, and making it second best in TMB2011. The result support our claim that learning useful syntactic patterns from noisy tweets is possible and that relational syntactic features generated by our shallow syntactic tree kernel improve over a strong feature-based L2R baseline.

[1] http://disi.unitn.it/moschitti/Tree-Kernel.htm

Table 1: System performance (P@30, MAP; higher is better) and system rank (R; lower is better) for UvA's L2R system [2] (UvA), our re-implementation (UvA*), and a UVA* system using our STRUCT model (+STRUCT). We report on relative improvement (Impr) and statisical significance against UvA*.

Model	TMB2011			TMB2012		
	MAP	P@30	R	MAP	P@30	R
UVA	.3880	.4460	6	.2450	.3920	26
UVA*	.3845	.4456	6	.2467	.3870	28
+ STRUCT	**.3991**	**.4571**	**2**	**.2683**	**.4277**	**13**
Change	+3.8%$^{\triangle}$	+2.6%	+4	+8.8%$^{\blacktriangle}$	+10.5%$^{\blacktriangle}$	+15

Table 2 reports on the application of our syntax-aware re-ranker on participant systems. It has results for re-ranking runs of the best 30 systems from TMB2012 (based on their P@30 score) when we train our system using the top {10, 20, or 30} runs from TMB2011. Our re-ranker improves P@30 for all systems with a relative improvement ranging from several points up to 10%—about 5% on average. This is remarkable, given that the pool of participants in TMB2012 was large, and the top systems are therefore likely to be very strong baselines. We observe that our syntactic model has a precision-enhancing effect. In cases where MAP drops a bit it can be seen that our model sometimes lowers relevant documents in the runs. It is possible that our model favors tweets with a higher syntactic quality, and that it down-ranks tweets that contain less syntactic structure but are nonetheless relevant. This is an interesting direction for analysis in future work.

Looking at the improvement in absolute position in the official ranking (R), we see that, on average, using our re-ranker boosts the absolute position in the official ranking for top 30 systems by 7 positions. All in all, the results suggest that using syntactic features adds useful information to many state-of-the-art microblog search algorithms.

Finally, using aggregate scores from the best 10, 20 or 30 systems from TMB2011 does not reveal large differences, which suggests that our syntax-aware re-ranker is robust w.r.t. the exact retrieval models used in the training stage.

While improving the top systems from 2012 represents a challenging task, it is also interesting to assess the potential improvement for systems that ranked lower. For this purpose, we select 30 systems from the middle and the bottom of the official ranking. Table 3 summarizes the average improvement in P@30 for three groups of 30 systems each: top-30, middle-30, and bottom-30. We find that the improvement over underperforming systems is much larger than for stronger systems. In particular, for the bottom 30 systems, our approach achieves an average relative improvement of 20% in both MAP and P@30. These results further support our hypothesis that syntactic patterns automatically extracted and learned by our re-ranker can provide an additional benefit for learning to rank methods on microblog data.

4. CONCLUSIONS

To the best of our knowledge, this work is the first to study the utility of syntactic patterns for microblog retrieval. We propose an efficient way to encode tweets into linguistic structures and use kernels for automatic feature engineering and learning. Our experi-

Table 2: System performance on the top 30 runs from TMB2012, using the top 10, 20 or 30 runs from TMB2011 for training.

		TMB2012		TOP10			TOP20			TOP30		
#	runs	MAP	P@30	MAP	P@30	R%	MAP	P@30	R%	MAP	P@30	R%
1	hitURLrun3	.3469	.4695	.3307 (-4.7%)$^{\nabla}$	.4831 (2.9%)	0	.3378 (-2.6%)	.4864 (3.6%)$^{\triangle}$	0	.3328 (-4.1%)$^{\nabla}$	.4774 (1.7%)	0
2	kobeMHC2	.3070	.4689	.3029 (-1.3%)	.4740 (1.1%)	1	.3065 (-0.2%)	.4768 (1.7%)	1	.3037 (-1.1%)	.4768 (1.7%)	1
3	kobeMHC	.2986	.4616	.2956 (-1.0%)	.4706 (2.0%)	2	.2989 (0.1%)	.4734 (2.6%)	2	.2965 (-0.7%)	.4718 (2.2%)	2
4	uwatgclrman	.2836	.4571	.3010 (6.1%)$^{\blacktriangle}$	.4729 (3.5%)$^{\triangle}$	3	.3032 (6.9%)$^{\blacktriangle}$	.4729 (3.5%)$^{\blacktriangle}$	3	.2995 (5.6%)$^{\blacktriangle}$	.4712 (3.1%)$^{\triangle}$	3
5	kobeL2R	.2767	.4429	.2734 (-1.2%)	.4452 (0.5%)	0	.2785 (0.7%)	.4514 (1.9%)	0	.2744 (-0.8%)	.4463 (0.8%)	0
6	hitQryFBrun4	.3186	.4424	.3102 (-2.6%)	.4554 (2.9%)	1	.3145 (-1.3%)	.4582 (3.6%)$^{\triangle}$	2	.3118 (-2.1%)	.4554 (2.9%)	2
7	hitLRrun1	.3355	.4379	.3200 (-4.6%)$^{\nabla}$	.4508 (3.0%)	2	.3266 (-2.7%)	.4542 (3.7%)$^{\triangle}$	2	.3226 (-3.9%)$^{\nabla}$	.4525 (3.3%)	2
8	FASILKOM01	.2682	.4367	.2827 (5.4%)$^{\blacktriangle}$	.4548 (4.1%)$^{\blacktriangle}$	3	.2841 (5.9%)$^{\blacktriangle}$	.4525 (3.6%)$^{\blacktriangle}$	3	.2820 (5.2%)$^{\blacktriangle}$	.4531 (3.8%)$^{\blacktriangle}$	3
9	hitDELMrun2	.3197	.4345	.3090 (-3.4%)$^{\nabla}$	.4446 (2.3%)	4	.3142 (-1.7%)	.4458 (2.6%)	4	.3105 (-2.9%)	.4424 (1.8%)	4
10	tsqe	.2843	.4339	.2832 (-0.4%)	.4435 (2.2%)	5	.2865 (0.8%)	.4458 (2.7%)	5	.2836 (-0.3%)	.4441 (2.4%)	5
11	ICTWDSERUN1	.2715	.4299	.2873 (5.8%)$^{\blacktriangle}$	.4610 (7.2%)$^{\blacktriangle}$	7	.2885 (6.3%)$^{\blacktriangle}$	.4576 (6.4%)$^{\blacktriangle}$	7	.2862 (5.4%)$^{\blacktriangle}$	.4582 (6.6%)$^{\blacktriangle}$	7
12	ICTWDSERUN2	.2671	.4266	.2809 (5.2%)$^{\triangle}$	.4503 (5.6%)$^{\blacktriangle}$	7	.2808 (5.1%)$^{\triangle}$	.4508 (5.7%)$^{\blacktriangle}$	7	.2785 (4.3%)$^{\triangle}$	.4475 (4.9%)$^{\blacktriangle}$	7
13	cmuPrfPhrE	.3179	.4254	.3159 (-0.6%)	.4486 (5.5%)$^{\blacktriangle}$	8	.3190 (0.4%)	.4452 (4.7%)$^{\blacktriangle}$	8	.3172 (-0.2%)	.4469 (5.1%)$^{\blacktriangle}$	8
14	cmuPrfPhrENo	.3198	.4249	.3167 (-1.0%)	.4497 (5.8%)$^{\blacktriangle}$	9	.3201 (0.1%)	.4480 (5.4%)$^{\blacktriangle}$	9	.3179 (-0.6%)	.4486 (5.6%)$^{\blacktriangle}$	9
15	cmuPrfPhr	.3167	.4198	.3117 (-1.6%)	.4441 (5.8%)$^{\blacktriangle}$	10	.3154 (-0.4%)	.4407 (5.0%)$^{\blacktriangle}$	8	.3130 (-1.2%)	.4379 (4.3%)$^{\triangle}$	8
16	FASILKOM02	.2454	.4141	.2725 (11.0%)$^{\blacktriangle}$	.4497 (8.6%)$^{\blacktriangle}$	11	.2721 (10.9%)$^{\blacktriangle}$	.4497 (8.6%)$^{\blacktriangle}$	11	.2718 (10.8%)$^{\blacktriangle}$	.4508 (8.9%)$^{\blacktriangle}$	11
17	IBMLTR	.2630	.4136	.2734 (4.0%)$^{\triangle}$	.4424 (7.0%)$^{\blacktriangle}$	10	.2758 (4.9%)$^{\blacktriangle}$	.4412 (6.7%)$^{\blacktriangle}$	10	.2734 (4.0%)$^{\triangle}$	.4441 (7.4%)$^{\blacktriangle}$	10
18	otM12ihe	.2995	.4124	.2968 (-0.9%)	.4333 (5.1%)$^{\blacktriangle}$	7	.3015 (0.7%)	.4339 (5.2%)$^{\blacktriangle}$	7	.2969 (-0.9%)	.4322 (4.8%)$^{\blacktriangle}$	7
19	FASILKOM03	.2716	.4124	.2861 (5.3%)$^{\blacktriangle}$	.4407 (6.9%)$^{\blacktriangle}$	12	.2879 (6.0%)$^{\blacktriangle}$	.4469 (8.4%)$^{\blacktriangle}$	14	.2859 (5.3%)$^{\blacktriangle}$	.4452 (8.0%)$^{\blacktriangle}$	14
20	FASILKOM04	.2461	.4113	.2584 (5.0%)$^{\blacktriangle}$	.4362 (6.1%)$^{\blacktriangle}$	11	.2596 (5.5%)$^{\blacktriangle}$	.4322 (5.1%)$^{\blacktriangle}$	9	.2575 (4.6%)$^{\blacktriangle}$	.4294 (4.4%)$^{\blacktriangle}$	9
21	IBMLTRFuture	.2731	.4090	.2803 (2.6%)	.4384 (7.2%)$^{\blacktriangle}$	14	.2830 (3.6%)$^{\triangle}$	.4328 (5.8%)$^{\blacktriangle}$	10	.2808 (2.8%)	.4311 (5.4%)$^{\blacktriangle}$	10
22	uiucGSLIS01	.2445	.4073	.2574 (5.3%)$^{\triangle}$	.4271 (4.9%)$^{\triangle}$	10	.2612 (6.8%)$^{\blacktriangle}$	.4260 (4.6%)$^{\blacktriangle}$	9	.2575 (5.3%)$^{\blacktriangle}$	.4260 (4.6%)$^{\blacktriangle}$	9
23	PKUICST4	.2786	.4062	.2913 (4.6%)$^{\triangle}$	.4537 (11.7%)$^{\blacktriangle}$	18	.2931 (5.2%)$^{\blacktriangle}$	.4486 (10.4%)$^{\blacktriangle}$	18	.2909 (4.4%)$^{\triangle}$	.4514 (11.1%)$^{\blacktriangle}$	18
24	uogTrLsE	.2909	.4028	.2983 (2.5%)	.4282 (6.3%)$^{\blacktriangle}$	12	.3015 (3.6%)$^{\triangle}$	.4243 (5.3%)$^{\blacktriangle}$	9	.2977 (2.3%)	.4282 (6.3%)$^{\blacktriangle}$	9
25	otM12ih	.2777	.3989	.2807 (1.1%)	.4260 (6.8%)$^{\blacktriangle}$	12	.2839 (2.2%)	.4232 (6.1%)$^{\blacktriangle}$	10	.2810 (1.2%)	.4175 (4.7%)$^{\blacktriangle}$	10
26	ICTWDSERUN4	.1877	.3887	.1995 (6.3%)$^{\blacktriangle}$	.4136 (6.4%)$^{\blacktriangle}$	8	.1992 (6.1%)$^{\blacktriangle}$	.4164 (7.1%)$^{\blacktriangle}$	10	.1985 (5.8%)$^{\blacktriangle}$	.4164 (7.1%)$^{\blacktriangle}$	10
27	uwatrrfall	.2620	.3881	.2829 (8.0%)$^{\blacktriangle}$	.4158 (7.1%)$^{\blacktriangle}$	11	.2841 (8.4%)$^{\blacktriangle}$	.4136 (6.6%)$^{\blacktriangle}$	9	.2812 (7.3%)$^{\blacktriangle}$	.4136 (6.6%)$^{\blacktriangle}$	9
28	cmuPhrE	.2731	.3842	.2792 (2.2%)	.4130 (7.5%)$^{\blacktriangle}$	10	.2810 (2.9%)	.4164 (8.4%)$^{\blacktriangle}$	12	.2797 (2.4%)	.4136 (7.7%)$^{\blacktriangle}$	12
29	AIrun1	.2237	.3842	.2350 (5.1%)$^{\blacktriangle}$	.4085 (6.3%)$^{\blacktriangle}$	7	.2359 (5.5%)$^{\blacktriangle}$	.4056 (5.6%)$^{\blacktriangle}$	5	.2339 (4.6%)$^{\blacktriangle}$	.4102 (6.8%)$^{\blacktriangle}$	5
30	PKUICST3	.2118	.3825	.2320 (9.5%)$^{\blacktriangle}$	.4220 (10.3%)$^{\blacktriangle}$	15	.2324 (9.7%)$^{\blacktriangle}$	.4181 (9.3%)$^{\blacktriangle}$	14	.2318 (9.4%)$^{\blacktriangle}$	.4119 (7.7%)$^{\blacktriangle}$	14
	Average			2.4%	5.4%	7.7	3.3%	5.3%	7.3	2.4%	5.0%	7.1

Table 3: System performance for top, middle (mid), and bottom (btm) 30 systems from TMB2012 system ranking and relative improvements using our method trained on top 20 (TOP20) performing systems in TMB2011.

	TMB2012		TOP20	
band	MAP	P@30	MAP	P@30
top	.2794	.4209	.2876 (3.3%)	.4430 (5.3%)
mid	.2193	.3460	.2461 (12.2%)	.3906 (12.9%)
btm	.1332	.2636	.1626 (22.1%)	.3298 (25.1%)

mental findings show that our model: (i) improves in both MAP and P@30 when coupled with the features from a strong L2R baseline; (ii) provides a complementary source of features general enough to improve the best 30 systems from TMB2012; (iii) the performance gains are stable when we use run scores from the top 10, 20 or 30 best systems for learning; and (iv) the improvement becomes larger for underperforming systems achieving an average 20% of relative improvement in MAP and P@30 for bottom 30 systems.

Acknowledgments. This research was partially supported by the Google Europe Doctoral Fellowship Award 2013, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 288024 (LiMoSINe) and nr 312827 (VOX-Pol), the Netherlands Organisation for Scientific Research under nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, the Center for Creation, Content and Technology (CCCT), the QuaMerdes project funded by the CLARIN-nl program, the TROVe project funded by the CLARIAH program, the Dutch national program COMMIT, the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), the Netherlands eScience Center under number 027.012.105 the Yahoo! Faculty Research and Engagement Program, the Microsoft Research PhD program, and the HPC Fund.

REFERENCES

[1] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: An analysis using six European languages. In *SIGIR*, 2007.

[2] R. Berendsen, M. Tsagkias, W. Weerkamp, and M. de Rijke. Pseudo test collections for training and tuning microblog rankers. In *SIGIR*, 2013.

[3] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *ACL*, 2011.

[4] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, 2006.

[5] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the TREC-2011 microblog track. In *TREC*, 2011.

[6] A. Severyn and A. Moschitti. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*, 2012.

[7] A. Severyn, M. Nicosia, and A. Moschitti. Learning semantic textual similarity with structural representations. In *ACL*, 2013.

[8] A. Severyn, M. Nicosia, and A. Moschitti. Building structures from classifiers for passage reranking. In *CIKM*, 2013.

[9] I. Soboroff, I. Ounis, J. Lin, and I. Soboroff. Overview of the TREC-2012 microblog track. In *TREC*, 2012.