The University of Amsterdam at TREC 2012

Richard Berendsen Edgar Meij Daan Odijk
Maarten de Rijke Wouter Weerkamp

ISLA, University of Amsterdam
`http://ilps.science.uva.nl/`

Abstract: We describe the participation of the University of Amsterdam's ILPS group in the Knowledge Base Acceleration and Microblog tracks at TREC 2012.

1 Introduction

This year the Information and Language Processing Systems (ILPS) group of the University of Amsterdam participated in the Microblog and the Knowledge Base Acceleration (KBA) tracks. In this paper, we describe our participation for each of these tracks, in two largely independent sections: Section 2 on our KBA track participation and Section 3 on our work in the Microblog track. We detail the runs we submitted, present the results of the submitted runs, and, where possible, provide an initial analysis of these results. We conclude in Section 4.

2 Knowledge Base Acceleration

This year's single task at TREC KBA concerned entity linking on a stream of data. That is, given a target entity from a knowledge base and an incoming stream consisting of textual content such as web pages, news items, and social media content, generate a score for each item based on how pertinent it is to the target entity.

Recent advances have enabled a precise manner of analysis, where phrases occurring in documents are automatically linked to entries in a knowledge base [11]. This process is commonly known as *entity linking*. Entity linking facilitates advanced forms of searching and browsing in various domains and contexts. It can be used, for instance, to anchor the textual resources in background knowledge; authors or readers of a piece of text may find entity links to supply useful pointers [6, 9]. Another application can be found in search engines, where it is increasingly common to link queries to entities and present entity-specific overviews [1, 8].

Typical solutions to entity linking operate on static collections of documents [9]. In this year's TREC participation, we adapt our entity linking system to be able to operate in this dynamic setting. To this end, we have implemented an incremental learning algorithm that updates at each time step, thereby improving performance at each point in time. Our algorithm also caters for different levels of aboutness.

2.1 Learning to Link Entities

We base our submissions mainly on the approach detailed in [9]. This approach is a learning to rerank approach to improve precision on a recall-oriented baseline. In this section we first introduce our general machine learning framework and the features we use. We then zoom in on how we adapt these to account for the incremental nature of the document collection.

2.1.1 Machine Learning

For entity linking on document streams we build upon a two-step process that has been shown to perform well on generic entity linking, i.e., exhaustively identifying all possible links in a piece of text [9]. The goal of the first step is to provide a filter and identify whether or not a document d contains a mention of the entity of interest, e. To this end, we consider all surface forms for e and determine whether the document contains any of these surface forms. In the second step we determine the level of centrality of e in d by applying supervised machine learning, using the set of features listed in Section 2.1.2. Using a machine learning framework allows us to encode different levels of relevance into our learning algorithm. Filtering the documents to be used as input for the machine learning algorithm reduces the number of feature vectors that need to be created in the second step, decreasing the end-to-end runtime.

We cast the second step as a binary classification problem and use the classifier's confidence to rank each document with respect to an entity. We employ random forests (RF) as our classification algorithm [2]. RF is an ensemble-based decision tree classifier based on bagging, in which a learning algorithm is applied multiple times to a subset of the instances and the results averaged. In this case, for each iteration a bootstrap sample is taken and a full tree is constructed. For each node of the tree, m features are randomly selected to obtain the best split. This process reduces overfitting by averaging classifiers that are trained on different subsets of

$TF(r_e, f_d)$	Term frequency.
$POS_f(r_e, f_d)$	Position of first occurrence.
$POS_l(r_e, f_d)$	Position of last occurrence.
$SPR(r_e, f_d)$	Distance between the first and last occurrence.
$SIM(f_d, a_e)$	Asymmetric similarity between f_d and a_e.
$SIM(a_e, f_d)$	Asymmetric similarity between a_e and f_d.
$CTA(e, f_d)$	Number of times all context is found in the document (e.g., "music group" for Basic_Element_(music_group).
$CTS(e, f_d)$	Number of times some context is found in the document (e.g., "music" for Basic_Element_(music_group).

Table 1: List of features we use. f_d refers to the document fields, r_e to the representations of e, and a_e to the Wikipedia article associated with e; more details can be found in Section 2.1.2.

the data with the same underlying distribution. RF is also relatively insensitive to parameter settings, resistant to overfitting, and easily parallelizable. We set m to the square root of the number of features [5]. Besides m, RF has one additional parameter: the number of iterations which we set to $k = 1500$.

2.1.2 Features

Table 1 shows the features that we use. They are calculated on two distinct parts of each document, f_d: the title and the body. All feature values are normalized by the length of this part of the document, if applicable. Moreover, for the features *TF*, POS_f, POS_l, and *SPR*, we extract separate feature values for each of the different representations of the entity, r_e. These representations are obtained from the Wikipedia article associated with e and include (i) the title, (ii) the text of the incoming anchors, (iii) the text of any incoming redirect links, and (iv) the union of these three. So, for instance for the feature *TF*, we actually have eight feature values, one for each combination of representation and document field. This yields a total of 48 features.

The *SIM* feature indicates the similarity between the entity and the document and is determined using the KL-divergence (KL-div) between the language models of the entity and the document:

$$\text{KL-div}(\theta||\theta') = \sum_{t \in \mathcal{V}} P(t|\theta) \log \frac{P(t|\theta)}{P(t|\theta')}, \quad (1)$$

where $t \in \mathcal{V}$ denotes a term in the shared vocabulary; each language model is estimated using a multinomial distribution over unigrams. To avoid zero frequency issues, we apply additive smoothing:

$$P(t|\theta_d) = \frac{\delta + n(t,d)}{\delta + \sum_{t'} n(t',d)}, \quad (2)$$

Run	Cutoff	Prec.	Recall	F	SU
Minimum relevance level "central"					
Baseline	0	0.1974	**0.9727**	0.2902	0.1339
Learning	181	0.3029	0.6130	0.3651	0.2565
IncLearnT1	98	0.2642	0.6910	0.3570	0.2074
IncLearnT5	199	0.3103	0.5819	0.3588	0.2687
IncLearnT10	191	**0.3152**	0.5782	**0.3681**	**0.2746**
Minimum relevance level "relevant"					
Baseline	0	0.5396	0.9806	0.6414	0.6044
Learning	3	**0.5482**	0.9191	**0.6448**	**0.6144**
IncLearnT1	0	0.5396	**0.9811**	0.6416	0.6047
IncLearnT5	0	0.5396	**0.9811**	0.6416	0.6047
IncLearnT10	0	0.5396	**0.9811**	0.6416	0.6047

Table 2: Results for TREC KBA. For each run the optimal cutoff is indicated; this is determined by considering the highest value for the F-measure for that run.

and set $\delta = 1$. Here, $n(t,d)$ denotes the count of term t in d. Note that KL-div is an asymmetric measure, hence we calculate it twice.

2.1.3 Incremental Learning

As we have a set of training documents, we experiment with a number of incremental learning variants. Each incremental learning algorithm can be cast as a sequential prediction problem, where for each time step $i = 1, 2, \ldots$:

1. An unlabeled document d_i arrives.
2. We perform a prediction $\hat{y}_i$, with confidence c_i based on e and the current model m_i.
3. We assume the predicted labels are correct and add a subset of documents as training material with assumed label y_i according to some constraint.
4. Update the model m_{i+1}.

We vary the constraints for the last step in our experimental conditions. In all cases, we consider a semi-supervised scenario, were we start with set of training material with annotated labels and try to learn from newly seen material. Here we start with a model m_1 based on the 2011 training data and update the model m_{i+1} at each step using the predictions made on documents arriving at step i. We define each day to be a time step.

2.2 Runs

We preprocess the TREC KBA document collection as follows. We tokenize each document, lowercase all characters, and remove all diacritics. From each document we use the title, body, and url fields. We implement our algorithm on

Run	R-Prec	MAP	MRR	P@5	P@10
Minimum relevance level "central"					
Baseline	**0.4514**	0.4484	0.5766	0.4828	0.5069
Learning	0.4445	**0.4533**	**0.7031**	**0.6207**	**0.5897**
IncLearnT1	0.4363	0.4362	0.6449	0.5379	0.5310
IncLearnT5	0.4372	0.4411	0.6515	0.5586	0.5103
IncLearnT10	0.4317	0.4366	0.5999	0.5724	0.5586

Table 3: Results for TREC KBA, measured by rank based evaluation metrics.

Hadoop; the code can be found on GitHub.[1] All our submitted runs are automatic runs.

UvAbaseline Run based on simple lexical matching, i.e., only applying the first step. This considers all surface forms for e and determines whether each document contains any of these surface forms: more frequent matches yield a higher score.

UvALearning Run that builds upon UvAbaseline and applies machine learning using the 2011 training documents to perform predictions.

UvAIncLearnT25 Incremental learning; add the top-k most probable documents for each class at time step i, where $k = 25$.

UvAIncLearnT50 Same as UvAIncLearnT25, with $k = 50$.

UvAIncLearnLow Incremental learning; We consider the confidence of the machine learning algorithm and include d_i only if the confidence of the machine learning algorithm is above a certain threshold for each class.

UvAIncLearnHigh Same as UvAIncLearnLow, with a higher threshold.

Note that we discovered a bug in our official submission files and the performance of our submitted runs is therefore not indicative of the performance of our system. Furthermore, we trained a single classifier for all entities, whereas our approach calls for training machine learning models on a per-entity basis. In the next section, we report on the results of the repaired version.

2.3 Results

We found that the threshold-based runs (UvAIncLearnLow and UvAIncLearnHigh) did not perform significantly different from the top-k runs (UvAIncLearnT25 and UvAIncLearnT50). We therefore do not include the former in our analysis, but discuss different, bug-fixed variants of the latter instead (see above).

[1]See `https://github.com/ejmeij/trec-kba`.

Run	Cutoff	Prec.	Recall	F	SU
Social					
Baseline	0	0.2092	**0.8708**	0.3025	0.1687
Learning	36	0.2306	0.7025	0.3122	0.1872
IncLearnT1	95	0.2455	0.5498	0.3119	0.2116
IncLearnT5	15	0.2374	0.6578	0.3138	0.1870
IncLearnT10	73	**0.2528**	0.5725	**0.3159**	**0.2180**
News					
Baseline	0	0.2346	**0.9013**	0.3163	0.2077
Learning	178	**0.3609**	0.6181	**0.4107**	0.3529
IncLearnT1	84	0.2906	0.6978	0.3736	0.2760
IncLearnT5	76	0.3110	0.7021	0.3892	0.2912
IncLearnT10	159	0.3490	0.6095	0.4023	**0.3674**
Linking					
Baseline	0	0.2064	**0.8807**	0.2888	0.1684
Learning	224	**0.2891**	0.5191	0.3046	**0.2640**
IncLearnT1	81	0.2345	0.6299	**0.3071**	0.2096
IncLearnT5	35	0.2255	0.6997	0.3069	0.1988
IncLearnT10	135	0.2472	0.5411	0.2983	0.2479

Table 4: Results per genre for TREC KBA. For each run the optimal cutoff is indicated; this is determined by considering the highest value for the F-measure for that run.

Table 2 shows the main results for our TREC KBA participation. We observe that our machine learning approach improves precision over the baseline, resulting at an improved highest F-measure. Incremental learning has a positive effect on performance in terms of F-measure. For the "central" relevance level, this is mainly due to a higher precision; recall at the highest F-measure is lower. Figure 1 shows the performance of the IncLearnT10 run at varying cutoff levels. We observe that precision remains quite constant over all cutoff levels, while recall increases as expected. This results in a highest F-measure at a rather low cutoff of 191. Overall, precision is fairly low, which is mainly due to the fact that we approach the task as a ranking problem (instead of classification). As such, we do not "classify" many documents as being non-relevant.

Table 3 shows the results of our runs as measured using rank-based evaluation measures. We observe a similar pattern as for precision, recall, F-measure, and SU, with our machine learning approach improving on most rank-based measures. For most measures, incremental learning is not able to improve on the static machine learning approach.

2.4 Per Genre

Next, we consider the performance for each genre (news, social, linking) separately. That is, we split up the "central" assessments based on the genre associated with each document. Table 4 shows the results. We observe that we obtain the best scores on all four measures in the news genre. Interestingly, in the news and linking genre, incremental learning

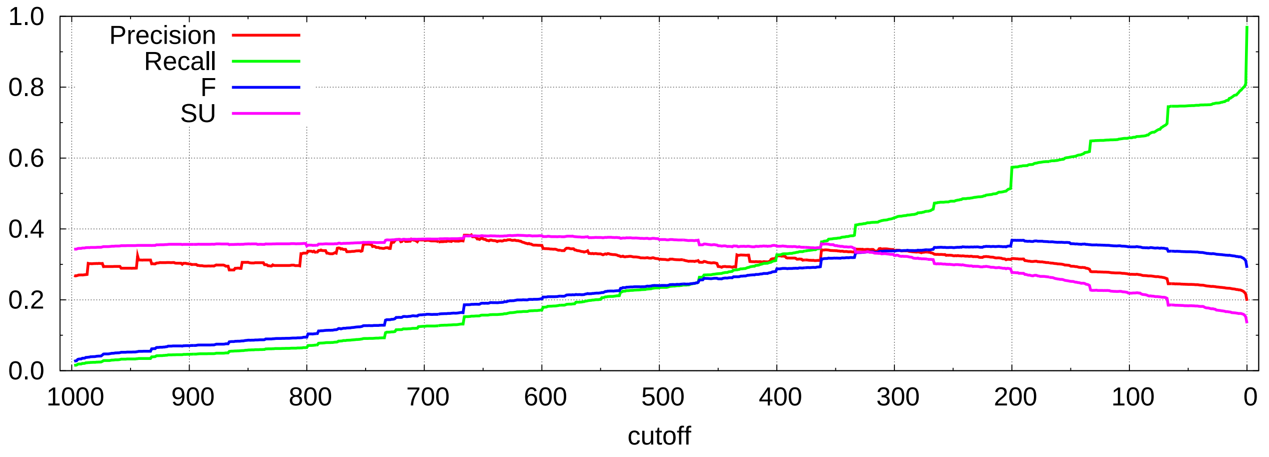

Figure 1: Performance changes over a varying cutoff for the TREC KBA IncLearnT10 run with minimum relevance level "central."

does not improve precision for the optimal cutoff, where it does improve in the social genre.

2.5 Per Day

Figure 2 shows the performance of each run in terms of MAP on each day in the test set, determined using the minimum relevance level "central." This plot shows that no run outperforms any other overall. We also observe a pattern of peaks that seems to be consistent with the number of documents in the collection for that particular day.

3 Microblog

This year's TREC Microblog track consists of two tasks: real-time ad-hoc and filtering. We participated only in the real-time ad-hoc task. Due to technical problems we could only submit one official run for this track (UvAFilter, see below), but in this section we also discuss the two runs that we originally planned to submit (UvAtrain2011 and UvAtrainGen). We discuss the following three runs in this section:

UvAFilter A baseline approach that filters out tweets based on a set of heuristics.

UvAFilterExp Baseline retrieval run with query expansion based on UvAFilter.

UvAtrain2011 A learning to rank approach trained on the TREC 2011 Microblog track topics.

UvAtrainGen The same learning to rank approach as UvAtrainGen, but trained on a generate pseudo test collection.

3.1 Preprocessing

For all our runs we use a preprocessed version of the corpus. First, we discard non-English tweets using a language identification method for microblogs [3]. We then remove exact duplicates and keep the oldest of each duplicate set, and finally, we discard retweets, unless there are added comments. For each tweet we remove punctuation and stopwords using a collection-based stopword list. After preprocessing our collection has just over four million tweets, roughly 25% of the raw collection.

3.2 UvAFilter

Based on observations from last year's Microblog track, we follow a heuristics-based approach for our baseline run. Starting from a retrieved set of tweets, constructed using Indri[2] and queries rewritten using a Markov random field model for term dependencies [10], we only keep tweets in our final result list if they match the following criteria: A tweet should (i) contain a URL, (ii) not contain a mention to another user ("@user . . . "), and (iii) does not refer to first or second person pronouns ("you," "me," "I," . . .).

The intuition behind the three filtering criteria is that a tweet only becomes "interesting" once it refers to more information besides the 140 characters of the tweets itself (the URL criterium), if it is not part of a conversation or directed as another user (the mention criterium and the second person pronouns), and finally, if it is not talking about personal opinions or experiences (the first person pronouns).

3.3 UvAFilterExp

A potential problem with query expansion is topic drift and the inclusion of non-informative terms from highly ranked

[2]http://www.lemurproject.com

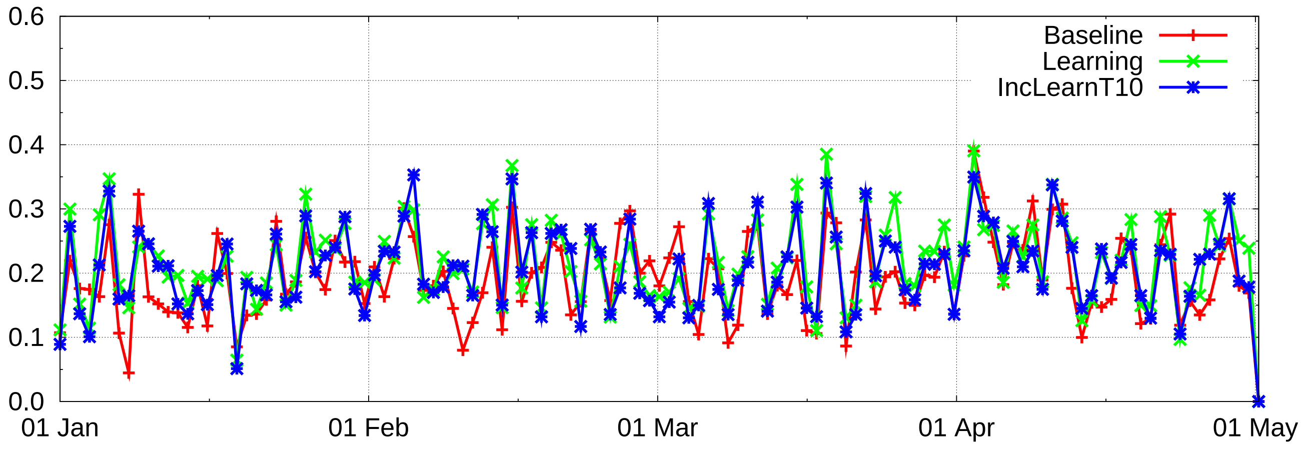

Figure 2: Performance changes in terms of MAP over each day with minimum relevance level "central."

documents. In this run we try to counter this by using the strictly filtered UvAFilter run as our input set of tweets for query expansion. We use a naive expansion technique that takes the top K tweets for each topic and return only those terms that appear more than N times in this set of tweets. We apply this method on our preprocessed collection of tweets, which does not contain stopwords.

We use the following settings: $K = 20$ and $N = 4$. As an example, Table 5 shows three example topics that improve over UvAFilter and one topic that drops in performance. For each topic we show the original query and the selected expansion terms.

3.4 UvAtrain2011

For microblog search, learning to rank (LTR) is a natural approach to take into account features such as the existence of a link, a hashtag, the recency of a tweet, the authority of the user, and so on. Training an LTR system requires good quality training material, the more the better. A natural training set are the TREC 2011 Microblog track topics. Our LTR run UvAtrain2011 run trains on these topics. We describe the details of this run below.

As features we use:

Query-document features Five Indri[2] runs; Five Terrier[3] runs; The number of rankers that retrieved a tweet; The {min, max, avg, median} reciprocal rank of a tweet; The recency of a tweet.

Query features Query clarity, as described in [4].

Tweet features Presence of a link, number of user mentions, tweet length, capitalization [14], density [7], is the message a direct message.

The parameters of Indri and Terrier retrieval runs were first tuned on the TREC 2011 microblog topics.

[3]`http://terrier.org/`

Learning to rank We linearly normalized all features. We used a pairwise SVM based learner [13], and set it to learn a linear model, optimizing the ROC area under curve. The regularization parameter λ was set to 0.1, and 100,000 iterations were performed.

3.5 UvAtrainGen

Training material for microblog search systems is expensive, and non-trivial to obtain. Therefore we explore the possibility of mining a collection of tweets for training material. We use hashtags for this, and are able to generate a pseudo test collection for microblog search. This pseudo test collection consists of a set of timestamped queries, with for each query a set of relevant tweets with a publication date prior to the timestamp. In the next paragraphs we describe how this pseudo test collection was generated. The UvAtrainGen run uses the same LTR pipeline as the UvAtrain2011 run, but it is trained on a generated pseudo test collection. Another difference is that the Indri and Terrier retrieval runs (which are used as features for the LTR run) are now tuned on the pseudo test collection as well.

Selecting hashtags We selected hashtags that appeared in at least fifty tweets in the TREC 2011 Microblog track collection.

Generating timestamps For each hashtag, the publication dates of its associated tweets form a time series. We set the timestamp for the hashtag to the time of the first peak in this series. We discard hashtags that have less than fifty hashtags before the timestamp.

Keeping interesting tweets We rank all tweets in our collection that contain one or more hashtags by their *interestingness*. We think of a tweet as interesting if it could be relevant to a query. We estimate interestingness by learning

Topic	Query	Expansion terms
55	berries and weight loss	been, fat, lose, amazon, berry, calories, acai, diets
86	Joanna Yeates murder	man, jo, charged, accused, remanded, tabak, vincent
109	Gasland	gas, oscar
65	Michelle Obama's obesity campaign	first, green, lady, weight, ap, loss, major, insurance, rate, role, response, plays, atlanta, rising, oprah, childhood, crescent

Table 5: Example topics and their selected query expansion terms. Topics 55, 86, and 109 improved over the baseline on most metrics, topic 65 dropped in performance.

from tweets that we know were relevant to some query: the union of all relevant tweets for the TREC 2011 Microblog track queries. After ranking the tweets, we keep the top fifty percent of them. We discard hashtags that have less than fifty interesting tweets.

Generating a query For each hashtag, we compare the set of associated tweets (T_h) with the collection of all tweets (T). We rank terms by a log-likelihood ratio test statistic. Terms of which the term frequency in T_h is most significantly different from the term frequency in T are ranked at the top. The top ten terms from this list form the query for the hashtag h [12]

3.6 UvA Microblog results

The results for our microblog runs are listed in Table 6. We find similar patterns for both relevance levels in that the two learning to rank runs (UvAtrain2011 and UvAtrainGen) perform almost identical. This shows that automatically constructing training data for this task is feasible and that a large set of human-annotated queries and tweets is not required.

The second finding is that our naive filtering baseline is a good choice when it comes to query expansion. The UvAFilter run itself shows strong performance on early precision (P10), which leads to an improved set of tweets from which to select expansion terms. This in turn results in improved performance of the UvAFilterExp run over both its baseline and thelearning to rank runs on most metrics. We should note, however, that differences are small.

Run	MAP	P30	P20	P10
Depth: 1000, minimum relevance level 1				
UvAFilter	0.1658	0.3384	0.3839	0.4627
UvAFilterExp	0.1852	**0.3627**	**0.3992**	**0.4644**
UvAtrain2011	**0.2153**	0.3514	0.3847	0.4390
UvAtrainGen	0.2057	0.3475	0.3890	0.4288
Depth: 1000, minimum relevance level 2				
UvAFilter	0.1385	0.1774	0.2085	**0.2780**
UvAFilterExp	**0.1561**	**0.1932**	**0.2161**	0.2644
UvAtrain2011	0.1524	0.1831	0.2076	0.2508
UvAtrainGen	0.1501	0.1836	0.2110	0.2542

Table 6: Results for the TREC 2012 Microblog track tealtime ad-hoc task.

4 Conclusion

In this paper we have described the participation of the University of Amsterdam's ILPS group at TREC 2012. We have participated in two tracks, Knowledge Base Acceleration (KBA) and Microblog. For the KBA track, we find that our machine learning-based approaches outperform the baseline. For the Microblog track our main observations are (i) that automatic construction of training data for a learning to rank approach is equally helpful as using a human-annotated dataset, (ii) that strict heuristics-based filtering leads to improved early precision, and (iii) that using this filtered run as basis for query expansion leads to best overall performance.

5 Acknowledgments

This research is supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 258191 (PROMISE Network of Excellence) and 288024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.814, 612.061.815, 640.004.802, 380-70-011, 727.-011.005, 612.001.116, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the WAHSP and BILAND projects funded by the CLARIN-nl program, the Dutch national programm COMMIT and by the ESF Research Network Program ELIAS.

6 References

[1] Balasubramanian, N. and Cucerzan, S. (2010). Topic pages: An alternative to the ten blue links. In *ICSC '10*.

[2] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[3] Carter, S., Weerkamp, W., and Tsagkias, M. (2012). Microblog language identification: overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, pages 1–21.

[4] Cronen-Townsend, S. and Croft, W. (2002). Quantifying query ambiguity. In *Proceedings of the second international conference on Human Language Technology Research*, pages 104–109. Morgan Kaufmann Publishers Inc.

[5] Hastie, T., Tibshirani, R., and Friedman, J. H. (2003). *The Elements of Statistical Learning*. Springer.

[6] He, J., de Rijke, M., Sevenster, M., van Ommering, R., and Qian, Y. (2011). Generating links to background knowledge: a case study using narrative radiology reports. In *CIKM '11*.

[7] Lee, G. G., Seo, J., Lee, S., Jung, H., hyun Cho, B., Lee, C., Kwak, B.-K., Cha, J., Kim, D., An, J., Kim, H., and Kim, K. (2001). SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *TREC 2001*, pages 442–451.

[8] Meij, E., Bron, M., Hollink, L., Huurnink, B., and de Rijke, M. (2011). Mapping queries to the Linking Open Data cloud: A case study using DBpedia. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):418 – 433.

[9] Meij, E., Weerkamp, W., and de Rijke, M. (2012). Adding semantics to microblog posts. In *WSDM '12*.

[10] Metzler, D. and Croft, W. B. (2005). A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 472–479.

[11] Mihalcea, R. and Csomai, A. (2007). Wikify!: Linking documents to encyclopedic knowledge. In *CIKM '07*.

[12] Rayson, P. and Garside, R. (2000). Comparing corpora using frequency profiling. In *Proceedings of the workshop on Comparing Corpora*, pages 1–6. Association for Computational Linguistics.

[13] Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *ICML '12*, pages 807–814. ACM.

[14] Weerkamp, W. and de Rijke, M. (2008). Credibility improves topical blog post retrieval. In *Proceedings of ACL-08: HLT*, page 923931, Columbus, Ohio. Association for Computational Linguistics, Association for Computational Linguistics.