The University of Amsterdam (ISLA) at INEX 2009

Krisztian Balog, Jiyin He, Marc Bron, Maarten de Rijke, and Wouter Weerkamp

ISLA, University of Amsterdam, Science Park 107, 1098 XG Amsterdam, The Netherlands
{k.balog, j.he, m.bron, derijke, w.weerkamp}@uva.nl

Abstract. We describe our participation in the INEX 2009 Entity Ranking and Link-the-Wiki tracks. We provide a detailed account of the ideas underlying our approaches to these tasks.

1 Introduction

This year the Intelligent Systems Lab Amsterdam at the University of Amsterdam participated in two INEX tracks: Entity Ranking and Link-the-Wiki.

For the Entity Ranking track our main emphasis was to evaluate a recently proposed probabilistic framework for entity retrieval that explicitly models category information in a theoretically transparent manner [2]. Information needs and entities are represented as a tuple: a term-based model plus a category-based model, both characterized by probability distributions over words. Ranking of entities is then based on similarity to the query, measured in terms of similarity between probability distributions. In our participation, our focus is on two core steps: query modeling and query model expansion. Moreover, we seek to answer how well parameter settings trained on the 2007 and 2008 editions of the Entity Ranking track perform on this year's setup.

In our participation in the Link-the-Wiki track our main aim was to explore the effectiveness of learning methods and learning materials for automatic generation of outgoing links. We experimented with two types of learning approaches: a classification-based approach and a ranking-based approach. We train the classifier as well as the ranker on two different versions of Wikipedia collections.

In this paper, we describe our participation for the tracks mentioned above, in two largely independent sections: Section 2 is devoted to our entity ranking track participation and Section 3 is devoted to our work in the link-the-wiki track. We conclude in Section 4.

2 Entity Ranking

In this section we present a probabilistic retrieval framework for the two tasks that have been formulated within the Entity Ranking track. In the *entity ranking* task we are given a query (q) and a set of target categories (C) and have to return entities. For *list completion* we need to return entities given a query (q), a set of similar entities (E), and (optionally also) a set of target categories (C).

Balog et al. [2] recently proposed a probabilistic retrieval model for entity search, in which term-based and category-based representations of queries and entities are effectively integrated. With the exception of the formula used for weighting terms for query expansion, we present the original approach unchanged.

The remainder of this section is organized as follows. In §2.1 we introduce our retrieval model, followed by the discussion of entity and query models in §2.2 and §2.3, respectively. We discuss our submitted runs in §2.4.

2.1 Modeling Entity Ranking

We rank entities e according to their probability of being relevant given the query q: $P(e|q)$. Instead of estimating this probability directly, we apply Bayes' rule and rewrite it to:

$$P(e|q) \propto P(q|e) \cdot P(e), \tag{1}$$

where $P(q|e)$ expresses the probability that query q is generated by entity e, and $P(e)$ is the *a priori* probability of e being relevant, i.e., the entity prior.

Each entity is represented as a pair: $\theta_e = (\theta_e^T, \theta_e^C)$, where θ_e^T is a distribution over terms and θ_e^C is a distribution over categories. Similarly, the query is also represented as a pair: $\theta_q = (\theta_q^T, \theta_q^C)$, which is then (optionally) refined further, resulting in an expanded query model that is used for ranking entities.

The probability of an entity generating the query is estimated using a mixture model:

$$P(q|e) = \lambda \cdot P(\theta_q^T|\theta_e^T) + (1-\lambda) \cdot P(\theta_q^C|\theta_e^C), \tag{2}$$

where λ controls the interpolation between the term-based and category-representations. The estimation of $P(\theta_q^T|\theta_e^T)$ and $P(\theta_q^C|\theta_e^C)$ requires a measure of the difference between two probability distributions. Here, we opt for the Kullback-Leibler divergence—also known as the relative entropy. The term-based similarity is estimated as follows:

$$P(\theta_q^T|\theta_e^T) \propto -KL(\theta_q^T||\theta_e^T) = -\sum_t P(t|\theta_q^T) \cdot \frac{P(t|\theta_q^T)}{P(t|\theta_e^T)}, \tag{3}$$

where the probability of a term given an entity model ($P(t|\theta_e^T)$) and the probability of a term given the query model ($P(t|\theta_q^T)$) remain to be defined. Similarly, the category-based component of the mixture in Eq. 2 is calculated as:

$$P(\theta_q^C|\theta_e^C) \propto -KL(\theta_q^C||\theta_e^C) = -\sum_c P(c|\theta_q^C) \cdot \frac{P(c|\theta_q^C)}{P(c|\theta_e^C)}, \tag{4}$$

where the probability of a category according to an entity's model ($P(c|\theta_e^C)$) and the probability of a category according to the query model ($P(c|\theta_q^C)$) remain to be defined.

2.2 Modeling Entities

Term-based representation To estimate $P(t|\theta_e^T)$ we smooth the empirical entity model with the background collection to prevent zero probabilities. We employ Bayesian

smoothing using Dirichlet priors which has been shown to achieve superior performance on a variety of tasks and collections [10, 6] and set:

$$P(t|\theta_e^T) = \frac{n(t,e) + \mu^T \cdot P(t)}{\sum_t n(t,e) + \mu^T}, \tag{5}$$

where $n(t,e)$ denotes the number of times t occurs in the document, $\sum_t n(t,e)$ is the total number of term occurrences, i.e., the document length, and $P(t)$ is the background model (the relative frequency of t in the collection). Since entities correspond to Wikipedia articles, this representation of an entity is identical to constructing a smoothed document model for each Wikipedia page, in a standard language modeling approach [9, 5]. Alternatively, the entity model can be expanded with terms from related entities, i.e., entities sharing the categories or entities linking to or from the Wikipedia page [3]. To remain focused, we do not explore this direction here.

Category-based representation Analogously to the term-based representation, we smooth the maximum-likelihood estimate with a background model. We employ Dirichlet smoothing, and use the parameter μ^C to avoid confusion with μ^T:

$$P(c|\theta_e^C) = \frac{n(c,e) + \mu^C \cdot P(c)}{\sum_c n(c,e) + \mu^C}. \tag{6}$$

In Eq. 6, $n(c,e)$ is 1 if entity e is assigned to category c, and 0 otherwise; $\sum_c n(c,e)$ is the total number of categories to which e is assigned; $P(c)$ is the background category model and is set using a maximum-likelihood estimate:

$$P(c) = \frac{\sum_e n(c,e)}{\sum_c \sum_e n(c,e)}, \tag{7}$$

where $\sum_c \sum_e n(c,e)$ is the number of category-entity assignments in the collection.

Entity priors By default, we use uniform entity priors, i.e., all pages in the collection are equally likely to be returned. Additionally, we experiment with priors that reward pages that are known to belong to entities; we use the 2007 and 2008 topic sets for setting the priors.

2.3 Modeling Queries

In this subsection we introduce methods for estimating and expanding query models. This boils down to estimating the probabilities $P(t|\theta_q^T)$ and $P(c|\theta_q^C)$ as discussed in §2.1.

Term-based representation The term-based component of the baseline query model is defined as follows:

$$P(t|\theta_q^T) = P_{bl}(t|\theta_q^T) = \frac{n(t,q)}{\sum_t n(t,q)}, \tag{8}$$

where $n(t,q)$ stands for the number of times term t occurs in query q.

The general form we use for expansion is a mixture of the baseline (subscripted with *bl*) defined in Eq. 8 and an expansion (subscripted with *ex*):

$$P(t|\theta_q^T) = (1 - \lambda^T) \cdot P_{bl}(t|\theta_q^T) + \lambda^T \cdot P_{ex}(t|\theta_q^T). \tag{9}$$

Given a set of feedback entities FB, the expanded query model is constructed as follows:

$$P_{ex}(t|\theta_q^T) = \frac{P_{K_T}(t|FB)}{\sum_{t'} P_{K_T}(t'|FB)}, \tag{10}$$

where $P_{K_T}(t|FB)$ is estimated as follows. First, $P(t|FB)$ is computed according to Eq. 11. Then, the top K_T terms with the highest $P(t|FB)$ value are taken to form $P_{K_T}(t|FB)$, by redistributing the probability mass, in proportion to their corresponding $P(t|FB)$ values.

$$P(t|FB) = \frac{1}{|FB|} \sum_{e \in FB} \frac{s(t,e)}{\sum_t s(t,e)} \tag{11}$$

and

$$s(t,e) = \log \frac{n(t,e)}{P(t) \cdot \sum_t n(t,e)}, \tag{12}$$

where $\sum_t n(t,e)$ is the total number of terms, i.e., the length of the document corresponding to entity e. (This is the same as the *EXP* query model generation method using example documents from [1], with the simplification that all feedback documents are assumed to be equally important.)

The set of feedback entities, FB, is defined in two ways: for the entity ranking task, it is the top N relevant entities according to a ranking obtained using the initial (baseline) query. For the list completion task, the set of example entities provided with the query are used as the feedback set ($FB = E$).

Category-based representation Our baseline model uses the keyword query (q) to infer the category-component of the query model (θ_q^C), by considering the top N_c most relevant categories given the query; relevance of a category is estimated based on matching between the name of the category and the query, i.e., a standard language modeling approach on top of an index of category names.

$$P(c|\theta_q^C) = P_q(c|\theta_q^C) = \begin{cases} P(q|\theta_c) / \sum_{c \in C} P(q|\theta_c), & \text{if } c \in \text{top } N_c \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Note that this method does not use the category information provided with the query. To use target category information, we set $n(c,q)$ to 1 if c is a target category, and $\sum_c n(c,q)$ to the total number of target categories provided with the topic statement. Then, we put

$$P_c(c|\theta_q^C) = \frac{n(c,q)}{\sum_c n(c,q)}. \tag{14}$$

To combine the two methods (categories relevant to the query and categories provided as input), we put:

$$P(c|\theta_q^C) = \frac{1}{2} P_q(c|\theta_q^C) + \frac{1}{2} P_c(c|\theta_q^C). \tag{15}$$

(For the sake of simplicity, each model contributes half of the probability mass.)

Expansion of the category-based component is performed similarly to the term-based case; we use a linear combination of the baseline (either Eq. 13 or Eq. 15) and expanded components:

$$P(c|\theta_q^C) = (1 - \lambda^C) \cdot P_{bl}(c|\theta_q^C) + \lambda^C \cdot P_{ex}(c|\theta_q^C). \tag{16}$$

Given a set of feedback entities FB, the expanded query model is constructed as follows:

$$P_{ex}(c|\theta_q^C) = \frac{P_{K_C}(c|FB)}{\sum_{c'} P_{K_C}(c'|FB)}, \tag{17}$$

where $P_{K_C}(c|FB)$ is calculated similarly to the term-based case: first, $P(c|FB)$ is calculated according to Eq. 18 (where, as before, $n(c, e)$ is 1 if e belongs to c). Then, the top K_C categories with the highest $P(c|FB)$ value are selected, and their corresponding probabilities are renormalized, resulting in $P_{K_C}(c|FB)$.

$$P(c|FB) = \frac{1}{|FB|} \sum_{e \in FB} \frac{n(c, e)}{\sum_t n(c, e)}. \tag{18}$$

The set of feedback entities is defined as before (the top N entities obtained using blind relevance feedback for entity ranking, and the example entities E for list completion).

2.4 Runs

Parameter settings Using the 2007 and 2008 editions of the Entity Ranking track as training material, we set the parameters of our models as follows.

- Importance of the term-based vs. the category-based component (Eq. 2): $\lambda = 0.7$
- Number of categories obtained given the query (Eq. 13): $N_c = 15$
- Number of feedback entities: $N = 3$
- Number of feedback terms (Eq. 17): $K_T = 35$
- Weight of feedback terms (Eq. 9): $\lambda^T = 0.7$
- Number of feedback categories (Eq. 17): $K_C = \infty$ (not limited)
- Weight of feedback categories (Eq. 16): $\lambda^C = 0.3$

Entity ranking Table 1 summarizes the 4 runs we submitted for the entity ranking task.

List completion Table 2 summarizes the 6 runs we submitted for the list completion task.

RunID	Baseline		Expanded		Priors	Description
(UAmsISLA_ER_...)	$P(t\|\theta_q^T)$	$P(c\|\theta_q^C)$	$P(t\|\theta_q^T)$	$P(c\|\theta_q^C)$		
TC_ERbaseline	Eq. 8	Eq. 15	-	-	N	Baseline
TC_ERfeedback	Eq. 8	Eq. 15	Eq. 10	Eq. 17	N	Feedback
TC_ERfeedbackS	Eq. 8	Eq. 15	Eq. 10	Eq. 17	N	Feedback (selected topics*)
TC_ERfeedbackSP	Eq. 8	Eq. 15	Eq. 10	Eq. 17	Y	Priors (on top of previous run)

Table 1. Entity ranking runs. (*Topics that were helped by blind feedback on the 2007/2008 topic set.)

RunID	Baseline		Expanded		Priors	Description
(UAmsISLA_LC_...)	$P(t\|\theta_q^T)$	$P(c\|\theta_q^C)$	$P(t\|\theta_q^T)$	$P(c\|\theta_q^C)$		
TE_LCexpT	Eq. 8	Eq. 13	Eq. 10	-	N	Feedback (term-based)
TE_LCexpC	Eq. 8	Eq. 13	-	Eq. 17	N	Feedback (category-based)
TE_LCexpTC	Eq. 8	Eq. 13	Eq. 10	Eq. 17	N	Feedback (term- + category-based)
TE_LCexpTCP	Eq. 8	Eq. 13	Eq. 10	Eq. 17	Y	Priors (on top of previous run)
TEC_LCexpTCS	Eq. 8	Eq. 13/15	Eq. 10	Eq. 17	N	Feedback (selected topics*)
TEC_LCexpTCSP	Eq. 8	Eq. 13/15	Eq. 10	Eq. 17	Y	Priors (on top of previous run)

Table 2. List completion runs. (*Topics that were helped by using example entities on the 2007/2008 topic set do not use input category information (i.e., use Eq. 13 for constructing $P_{bl}(c|\theta_q^C)$); the remainder of the topics use the input category information (i.e., $P_{bl}(c|\theta_q^C)$ is estimated using Eq. 15).)

3 Link-the-Wiki

In this section, we describe our participation in the Link-the-Wiki (LTW) track. The aim of the LTW track is to automatically identify hyperlinks between documents. We only participated in the task of outgoing link generation within Wikipedia (A2B). In our experiments, we focus on exploring machine learning methods and learning material for link detection.

The main purpose of our experiments are two-fold. First, we want to test how our learning methods work on the LTW task, especially how the results learnt from Wikipedia ground truth would be judged by human assessors. On top of that, since the LTW task is defined as a ranking problem for recommendation purposes, we want to see how a learning to rank approach works as it directly optimizes the rankings instead of assigning binary decisions to candidate links as a classification method would do. Second, we trained our models with different versions of Wikipedia. The two versions used, namely Wikipedia 2008 and Wikipedia 2009, differ in the amount of articles as well as the amount of links. Presumably, the 2009 version contains more link information but is also more noisy in terms of missing target pages (as some pages are deleted as time passes by). We experiment with both collections so as to see the impact of the training material used.

Learning Stage	N-gram	N-gram-target	Target	N-gram-topic	Topic-target	1st-stage
Candidate targets ranking		x	x		x	
Candidate links ranking	x	x	x	x	x	x

Table 3. Features and their corresponding application in different learning stages.

3.1 A two-stage learning procedure

Following [8], we consider the linking task as a two stage procedure, namely *candidate target identification* and *link detection*. First, we extract all possible n-grams in a topic page, and train a link-detector to rank the potential target pages for each n-gram, which we refer to as *candidate target pages*.

We experiment with two types of learning methods, namely classification and learning to rank. For classification, we use SVM to classify the instances in both stages, and rank the results by the probability of an instance being positive. For our learning to rank approach, we use RankingSVM [4] to directly optimize the ranking of an instance. In the candidate target identification stage, we train a ranker to rank the target candidates for each n-gram and in the link detection stage a ranker is trained to rank the n-gram target pairs.

3.2 Features

We identify 6 types of feature for learning a preference relation between the candidate links. Table 3 specifies in which stage each type is used and Table 4 lists the features. Here, we discuss the motivations of using these features, as well as detail the formulation of some of the features.

N-gram features The n-gram features suggest how likely a given n-gram would be marked as an anchor text, without any other information such as its context in the topic page, which includes its length, IDF score, number of candidate targets associated with it, and its ALR (Anchor Likelihood Ratio) scores. IDF is calculated as

$$\log(\frac{|D|}{|\{d_i : ng \in d_i\}_{i=1}^{N}|}),$$

where ng is a n-gram, d_i is a page containing a this n-gram, and D is the total collection of Wikipedia pages. The ALR score can be interpreted as a model selection between two models, the anchor model and the collection model, from which a n-gram is generated. To calculate the probability of a n-gram being generated by either model, the maximum likelihood is used. Specifically, it is calculated as

$$ALR(ng) = \frac{|ng \in A|}{|A|} \cdot \frac{|C|}{|ng \in C|} \tag{19}$$

where A is the collection of all anchor texts in the Wikipedia collection and C is the Wikipedia collection. A large ALR value indicates that the n-gram is more likely to have been generated from the anchor model, i.e., this n-gram is more likely to be an anchor text than a common word sequence from the background collection.

N-gram - target features The n-gram - target features describe how well an n-gram and its corresponding candidate target page are related. On the assumption that each Wikipedia page is about a specific concept that is usually denoted by its title, the first feature we use is the match between an n-gram and the candidate target page. The second type of feature in this category consists of indicators of how likely a given n-gram ng and a candidate target page $ctar$ are linked, which is expressed by the following two scores: *RatioLink* and *RatioAnchor*. The former is the ratio between the number of times ng and $ctar$ are linked and the number of times $ctar$ is being linked as a target page in the collection. The latter, i.e., *RatioAnchor* is the ratio between the number of times ng and $ctar$ are linked and the number of times ng is used as an anchor text in the collection. Moreover, we adopt retrieval scores between the n-gram and the candidate target pages as features (n-gram as query), which is an obvious description of the relatedness of the two:

$$RatioLink(ng, ctar) = \frac{|link(ng, ctar)|}{|inlink(ctar)|} \tag{20}$$

$$RatioAnchor(ng, ctar) = \frac{|link(ng, ctar)|}{|ng \in A|} \tag{21}$$

Target features The target features are indicators of how likely a candidate target page alone would be linked with some anchor text in the collection. To this end we explore features such as counts of the inlinks and outlinks within the candidate target page, as well as the Wikipedia category information associated with it.

N-gram - topic features This type of feature describes the importance of the n-gram within its context, i.e., topic page. One would assume that an n-gram being selected as an anchor text should be somewhat important to the understanding of the whole topic page as well as being content-wise related. Here, we use the TFIDF score of the n-gram and its location within the topic page as an indication of the importance of a n-gram within a topic page.

Topic - target features The topic-target features describe the relatedness between a topic page and a candidate target page. One obvious feature is the similarity between the two pages. In addition, as a candidate target page itself is about a concept, we could measure how important is this concept, or in other words, how well is this concept being expressed in the topic page. We measure it by using the title of the candidate target page as query and calculate the retrieval score against the topic page.

First stage score Once the target ranking has been completed (during the first stage), we can get the ranking score for each candidate target, as well as their ranks. In the second stage, we select the top X candidate targets to construct the candidate links with their corresponding n-grams, where the scores and ranks from the first stage are used as features.

3.3 The LTW Runs

We submitted 5 runs for the LTW task, as specified in Table 5.

Note that we have a heuristic run UvAdR_LTWA2B_03 which does not a use learning method for link ranking, but only use RankingSVM for target identification. This run serves as a baseline for other machine learning based approaches. The heuristics used in this run, i.e., the *ALR* and *IDF* scores, however, are the features that are most close to the human intuitions, where *ALR* represents how likely a n-gram is involved in a link based on the observation of existing links and *IDF* represents the uncommonness of a n-gram.

4 Conclusions

We have described our approaches and submissions for this year's INEX participation.

For the Entity Ranking track, we submitted 4 runs for the entity ranking and 6 runs for the list completion tasks. Our main focus is on evaluating the effectiveness of our recently proposed entity retrieval framework on the new Wikipedia collection. In addition, we are interested in investigating whether parameter settings learned on prior editions of the track carry over to this year's setting.

As to the Link-the-Twiki track, we submitted 5 runs to the A2B outgoing links detection task. Our main focus is to explore the effectiveness of applying machine learning approaches for the task. Specifically, we experiment with two types of learning approaches, namely classification and learning to rank. On top of that, we also aim to evaluate the learning material for the task, where we use different sets of training data (based on different versions of Wikipedia).

Acknowledgements

This research was supported by the DAESO and DuOMAn projects carried out within the STEVIN programme which is funded by the Dutch and Flemish Governments (http://www.stevin-tst.org) under project numbers STE-05-24 and STE-09-12, and by the Netherlands Organisation for Scientific Research (NWO) under project numbers 640.001.501, 640.002.501, 612.066.512, 612.061.814, 612.061.815, 640.004.802.

Bibliography

[1] K. Balog, W. Weerkamp, and M. de Rijke. A few examples go a long way: constructing query models from elaborate query formulations. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 371–378, New York, NY, USA, 2008. ACM.

[2] K. Balog, M. Bron, and M. de Rijke. Category-based query modeling for entity search. In *32th European Conference on Information Retrieval (ECIR 2010)*, To appear, 2010.

[3] S. Fissaha Adafre, M. de Rijke, and E. Tjong Kim Sang. Entity retrieval. In *Recent Advances in Natural Language Processing (RANLP 2007)*. Borovets, Bulgaria, September 2007.

[4] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[5] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, New York, NY, USA, 2001. ACM.

[6] D. Losada and L. Azzopardi. An analysis on document length retrieval trends in language modeling smoothing. *Information Retrieval*, 11(2):109–138, 2008.

[7] D. Metzler and W. Croft. 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. ACM New York, NY, USA, 2005.

[8] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 509–518, New York, NY, USA, 2008. ACM.

[9] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, New York, NY, USA, 1999. ACM.

[10] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214, 2004.

N-gram features	
Length(ng)	Number of words contained in the n-gram
IDF(ng)	The IDF score of the n-gram
ALR(ng)	The ALR score of the n-gram, as detailed in Eq. 19
#Cand(ng)	Number of candidate target pages associated with the n-gram
N-gram - target features	
TitleMatch(ng, ctar)	Three values - 2: exact match; 1: partial match(i.e., either the title contains the n-gram, or the n-gram contains the title); 0: not match
RatioLink(ng, ctar)	Link ratio of the n-gram and the candidate target page, see Eq. 20
RatioAnchor(ng, ctar)	Anchor ratio of the n-gram and the candidate target page, see Eq. 21
Ret_uni(ng, ctar)	Retrieval score with unigram model, i.e., BM25 with default parameter settings
Ret_dep(ng, ctar)	Retrieval scores with dependency model, i.e., Markov Random Field model as described in [7]
Rank_dep(ng, ctar)	The rank of the target page with the dependency retrieval model
Target features	
#Inlinks(ctar)	Number of in-links contained in the candidate target page
#Outlinks(ctar)	Number of out-links contained in the candidate target page
#Categories(ctar)	Number of Wikipedia categories associated with the candidate target page
Gen(ctar)	Generality of the candidate target page as described in [8]
N-gram - topic features	
TFIDF(ng, topic)	The TFIDF score of the n-gram in the topic page
First(ng, topic)	Position of first occurrence of the n-gram in the topic page, normalized by the length of the topic page
Last(ng, topic)	Position of last occurrence of the n-gram in the topic page, normalized by the length of the topic page
Spread(ng, topic)	Distance between first and last occurrence of the n-gram in the topic page normalized by the length of the topic page
Topic-target features	
Sim(ctar, topic)	Cosine similarity between the candidate target page and the topic page
Ret_unigram(ctar, topic)	Retrieval score using the title of the candidate target page as query against the topic page; using BM25 as retrieval model
First stage scores	
score(ng, ctar)	The output of the ranker for the candidate target page given the n-gram
rank(ng, ctar)	The rank of the candidate target page according to the learnt ranker

Table 4. Features used for learning the preference relation, where ng: n-grams; C: collection; ctar: candidate target pages; topic: topic page.

RunID	Description
UvAdR_LTWA2B_01	Binary classification, trained on wiki08 (old and small)
UvAdR_LTWA2B_02	Ranking SVM, trained on wiki08
UvAdR_LTWA2B_03	A heuristic run, combine the ALR and IDF for link ranking, ignore numbers, but using rankingSVM for target ranking
UvAdR_LTWA2B_04	Binary classification, trained on wiki09
UvAdR_LTWA2B_05	ranking SVM, trained on wiki09

Table 5. Submitted runs