The University of Amsterdam (ILPS) at INEX 2008

Wouter Weerkamp, Jiyin He, Krisztian Balog, and Edgar Meij

University of Amsterdam, ISLA, Kruislaan 403, 1098SJ Amsterdam, The Netherlands,
{w.weerkamp,j.he,k.balog,e.j.meij}@uva.nl

Abstract. We describe our participation in the INEX 2008 Entity Ranking and Link-the-Wiki tracks. We provide a detailed account of the ideas underlying our approaches to these tasks. For the Link-the-Wiki track, we also report on the results and findings so far.

1 Introduction

This year the Information and Language Processing Systems (ILPS) group of the University of Amsterdam participated in two INEX tracks: Entity Ranking and Link-the-Wiki. For the Entity Ranking track our main emphasis was on developing a general language modeling framework to model the tasks at hand. The models that are developed for both task offer us plenty of opportunity to experiment with various ways of estimating components of the models. We describe the components of the models and the way we estimated these. Of main interest to us was the use of category information in the various components (e.g., by estimating entity similarity in list completion), combining different entity representations, and query modeling. In our participation in the Link-the-Wiki track our main aim was to explore different features that indicate links between Wikipedia pages, as well as to develop a generative approach to automatic link generation. We submitted runs designed to compare the influence of different features on link generation.

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

2 Entity Ranking

The Enitity Ranking track of this year's INEX features three tasks: *entity ranking*, *list completion*, and *entity relation search*. In our participation, we focus on the first and second tasks, leaving entity relation search for coming years. Both tasks (entity ranking and list completion) are aimed at retrieving entities from a semi-structured document collection. The document collection at hand is Wikipedia, and an entity by definition is a Wikipedia article.

The entity ranking task aims at retrieving entities (Wikipedia pages) given a certain topic and Wikipedia category: the goal is to identify the Wikipedia pages that are relevant given the topic and fit within the given category. The list completion task is slightly different from the previous task, and aims at adding entities of the same type to a small

sample seed set of entities. Again, we also have the topic and category available, but as additional information we get a list of examples: one or more entities that fulfill the constraints.

In our participation we use a generative language modeling approach to model both tasks, since this offers us a theoretically sound and understandable way of dealing with the problem at hand. A large portion of this paper is directed to the modeling of entity ranking and the estimation of the various components this model offers us. We submitted a total of six runs, again with a focus on the entity ranking task (four runs).

The remainder of this section introduces the modeling of the entity ranking task in Section 2.1, and of the list completion task in Section 2.2. Next, we discuss the estimation of the various components of both models in Section 2.3, and we conclude with some notes on the experimental setup and submitted runs in Sections 2.4 and 2.5, respectively. Since evaluation results are not available at the time of writing, we skip the results, analysis, and conclusions.

2.1 Modeling Entity Ranking

Entities are ranked by their probability of being relevant given a query q and a set of categories C, that is $P(e|q, C)$. We assume that q and C are conditionally independent, moreover, it is also assumed that each of the categories $c \in C$ are mutually independent. Formally:

$$\begin{aligned} P(e|q, C) &= P(e|q) \cdot P(e|C) \\ &= P(e|q) \cdot \prod_{c \in C} P(e|c). \end{aligned} \tag{1}$$

To estimate the probability of an entity given the query, $P(e|q)$, we apply Bayes' rule, then drop the denominator, $P(q)$, which is a constant for all entities, and thus, does not influence the ranking:

$$P(e|q) \propto P(q|e) \cdot P(e). \tag{2}$$

Here, $P(q|e)$ expresses the probability that q is generated by entity e, and $P(e)$ is the *a priori* probability of an entity being relevant (independent of the query). For the sake of simplicity, $P(e)$ is assumed to be uniform, and is not included in the equations from now onwards.

We infer an entity model for each entity e, such that the probability of a term given the entity model is $P(t|\theta_e)$. This model is then used to predict how likely the entity would produce query q. Each query term is assumed to be sampled identically and independently. Thus, the query likelihood is obtained by taking the product of the individual term probabilities across all terms in the query:

$$P(q|\theta_e) = \prod_{t \in q} P(t|\theta_e)^{n(t,q)}, \tag{3}$$

where $n(t, q)$ denotes the number of times t is present in q. Putting together our choices so far (Eqs. 1, 2, and 3) we obtain the following:

$$P(e|q, C) = \prod_{t \in q} P(t|\theta_e)^{n(t,q)} \cdot \prod_{c \in C} P(e|c). \tag{4}$$

For computational reasons, we move to the log domain, and use the following formula for ranking entities:

$$\log P(e|q,C) \propto \left(\sum_{t \in q} P(t|\theta_q) \cdot \log P(t|\theta_e) \right) + \sum_{c \in C} \log P(e|c). \tag{5}$$

Note that $n(t,q)$ has been replaced with $P(t|\theta_q)$, where θ_q is referred to as the *query model*. This allows us more flexible weighting of query terms. Three important components remain to be defined: the entity model θ_e, the query model θ_q, and the probability of an entity given a category $P(e|c)$. These will be introduced in the following sections.

2.2 Modeling List Completion

The list completion task is modeled similarly to the entity ranking task, with the addition that the probability of the entity is also conditioned on a set of example entities, E. We assume that example entities are conditionally independent from the query and the categories, as well as mutually independent from each other. Formally:

$$P(e|q,C,E) = P(e|q,C) \cdot \prod_{e' \in E} P(e|e'). \tag{6}$$

Again, we perform this computation in the log domain:

$$\log P(e|q,C,E) = \log P(e|q,C) + \sum_{e' \in E} \log P(e|e'). \tag{7}$$

The estimation of $P(e|q,C)$ has already been discussed in Section 2.1. A new component to be defined is the probability of an entity e given entity e': $P(e|e')$. In other words, this probability expresses the similarity of two entities.

2.3 Estimating the Components

In this section we detail how various components of the models introduced in the previous sections are estimated. Specifically, we discuss the implementation of the entity model θ_e, the query model θ_q, the probability of an entity given a set of categories $P(e|C)$, and finally, the probability of an entity given another entity $P(e|e')$.

Entity Model The entity is represented as a multinomial probability distribution over terms. To estimate $P(t|\theta_e)$ we smooth the empirical entity model with the background collection to prevent zero probabilities:

$$P(t|\theta_e) = (1 - \lambda_e) \cdot P(t|e) + \lambda_e \cdot P(t) \tag{8}$$

Since entities correspond to Wikipedia articles, this way of modeling an entity is identical to constructing a smoothed document model for each Wikipedia page. The choice of the smoothing parameter λ_e is discussed in Section 2.4.

Query Model Our baseline query model $P(t|\theta_q)$ is set to $n(t,q) \cdot |q|^{-1}$, where $n(t,q)$ is the number of occurrences of term t in query q, and $|q|$ is the query length. Essentially, the probability mass is distributed uniformly across query terms. Since this representation of the query is quite sparse, we would like to add more terms to the original query. By mixing new terms and original query terms, we end up with the following equation:

$$P(t|\theta_q) = (1 - \lambda_q) \cdot P(t|\hat{q}) + \lambda_q \cdot P(t|q), \tag{9}$$

where $P(t|\hat{q})$ is the probability of the term given the expanded query.

In [1] various methods are introduced for constructing expanded query models by sampling terms from a set of example documents (complementing the textual query). Based on the information provided with the topic statement, we have three straightforward ways of applying these methods to our current scenario, by (i) treating all entities belonging to the target categories as examples (both for entity ranking and list completion), (ii) employ a blind-relevance feedback approach, in which we perform a baseline run and look at the categories which are assigned to the 10 highest-ranked entities (category feedback for entity ranking), or (iii) using the example entities (only list completion). Specifically, we use the best performing method, maximum likelihood (ML), from [1] to estimate the expanded query model $P(t|\hat{q})$.

Entity-Categories Probability The probability of an entity e given a set of target categories C, $P(e|C)$, is computed as follows. Let cat(e) denote the set of categories e is assigned to. The overlap ratio between cat(e) and the set of target categories C is used as an estimate of $P(e|C)$:

$$P(e|C) = \frac{|\text{cat}(e) \cap C|}{|C|}, \tag{10}$$

where $|C|$ is the size of the set of target categories. We experiment further with this way of estimating $P(e|C)$, by introducing a parameter δ to control the weight of the overlap between the two sets C and cat(e) and dropping the term in the denominator:

$$P(e|C) = \delta \cdot |\text{cat}(e) \cap C|. \tag{11}$$

Based on initial experiments we set $\delta = 6$.

Further, we hypothesize that the target categories for each topic, as used in Eqs. 10 and 11, are not exhaustive. Therefore, in order to amend this set of categories, we apply a simple expansion strategy. We leverage the hierarchical structure of the Wikipedia categories and add the categories below each target and expanded category up to a certain depth. Based on preliminary experiments, we set the maximum depth to three.

Entity-Entity Probability Our model for the list completion task involves the estimation of the similarity between two entities. This is expressed as $P(e|e')$, the probability of an entity e given another entity e' (see Eq. 6). We estimate this probability based on set overlap between the categories assigned to each of the entities. To this end, we employ a standard set-based similarity measure, Dice's coefficient, calculated as follows:

$$P(e|e') = \frac{2 \cdot |\text{cat}(e) \cap \text{cat}(e')|}{|\text{cat}(e)| + |\text{cat}(e')|}, \tag{12}$$

where cat(e) and cat(e') are the set of categories assigned to entities e and e', respectively.

2.4 Experimental Setup

Document Representation Besides representing the entity (Wikipedia page) by its entire textual content (refered to as *full representation*), we opted for a second representation. Assuming that most valuable information on a Wikipedia page is presented at the beginning of the article, we select only the first paragraph of each article. This paragraph is the new representation of the entity (refered to as *paragraph representation*). For reasons of comparability, we use the full representation in almost all cases.

Document Preprocessing Document preprocessing consisted of removing stopwords only. Besides the "standard" English stopwords, we added several Wikipedia-specific stopwords to the stopword list (e.g. *disambiguation*, *category*, and *stub*).

Smoothing Parameter For the smoothing parameter λ_e in Eq. 8, we set λ_e equal to $\frac{|e|}{\beta+|e|}$, where $|e|$ is the length of the entity in number of terms. Essentially, the amount of smoothing is proportional to the length of the entity (and is like Bayes smoothing with a Dirichlet prior [2]). If there is very few content available in the entity then the model of the entity is more uncertain, leading to a greater reliance on the background probabilities. We set β to be the average entity length, i.e. $\beta = 409$ for the full representation and $\beta = 42$ for the paragraph representation.

Query Modeling Parameter For the construction of the new query model (Eq. 9), we need to set λ_q and decide on the number of terms in the new query. For the entity ranking task, in which we select our expansion terms from the category feedback approach, we set $\lambda_q = 0.5$ and select the 20 terms with the highest probability. For the list completion task we set $\lambda_q = 0.2$ and again select the top 20 terms to be included in the new query.

2.5 Submitted Runs

The following lists our six submitted runs and the configuration used for each run (note that all runs use the full representation, unless stated otherwise).

6_UAms_ER_T_baseline: Our baseline run using Eq. 3.

3_UAms_ER_T_overlap: Overlap run using Eq. 5; we estimate $P(e|C)$ as in Eq. 11, and use an expanded category set C up to depth three.

4_UAms_ER_T_cat-exp: Expanded overlap run; similar to run *3_UAms_ER_T_overlap*, except that we model the query according to Eq. 9, where expansion terms are selected using the category feedback method. We select the top 2 categories and use the entities within these categories as examples.

1_UAms_ER_T_mixture: Mixture run; we construct two runs using Eq. 5, one on the full representation and one on the paragraph representation. Each run estimates $P(e|C)$ as in Eq. 11, and uses an expanded category set C up to depth three (paragraph representation) or two (full representation). Both runs are combined using a linear rank combination with a weight of 0.1 for the paragraph representation and 0.9 for the full representation.

The remaining two runs are used for the list completion task:

5_UAms_LC_T_baseline: Our baseline run using Eq. 7; we model the query according to Eq. 9 and select expansion terms from the provided example entities.

2_UAms_LC_T_dice: Our overlap run; similar to run *5_UAms_LC_T_baseline*. We estimate $P(e|e')$ using Eq. 12 and $P(e|C)$ as in Eq. 11, and use an expanded category set C up to depth two.

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. The 2006 Wikipedia collection is used as the development and test data, which contains the ground truth of linked documents. This year's LTW track consists of two sub-tasks, namely, the file-to-file (f2f) link generation task and the anchor-text to Best Entry Point (BEP) link generation task. We participated in the f2f task and submitted three runs. The task is formulated as follows: a set of 6600 Wikipedia articles are randomly picked from the collection as topic pages; the participants are supposed to discover at most 250 incoming and 250 outgoing links between the topic pages and the rest of the collection. In the following sections, we introduce our approaches for both incoming links and outgoing links, followed by the description of the runs we submitted, as well as the experimental settings we applied to our runs.

3.1 Outgoing Links

Our approach to identifying outgoing links can be seen as a two-step procedure: anchor text detection (where to start a link) and target identification (which page should be linked). Although in the f2f task, the exact position of the anchor text is not required, the identified anchor texts strongly indicate the target pages to be linked.

Anchor Likelihood Ratio For anchor text detection, we introduce the anchor likelihood ratio measure (ALR) which we use to rank the n-grams in a text on being an anchor text. Since the Wikipedia articles are well-structured and interconnected, the existing links provide an indication of the patterns of linked pages. Mihalcea and Csomai [3] proposed the link probability measure which measures the likelihood of a word sequence being an anchor text by calculating the ratio between the number of times a word sequence is used as an anchor text and the number of times this word sequence occurs in the collection. The likelihood estimation is a reasonable measure and proved

to be useful. However, the value of the likelihood is a continuous number between 0 and 1, which needs a threshold to determine wether the word sequence is an anchor text or not. By modifying this measure, we try to model it without a magic threshold.

For a given word sequence w, we assume that it is sampled from two different underlying models: the anchor model θ_A and the background collection model θ_C. For selecting the model for generating the word sequence, we take the likelihood ratio between the two models, which is formulated as:

$$ALR_w(\theta_A||\theta_C) = \frac{P(w|\theta_A)}{P(w|\theta_C)} \tag{13}$$

Given Eq. 13, it is obvious that the larger the value of the likelihood ratio, the more likely it is that the word sequence w is generated by the anchor model. Particularly, when $ALR_w > 1$, it expresses that the anchor model is preferred over the collection model, and therefore we can obtain a non-magic threshold at $ALR_w = 1$.

Target page identification - Title Field Evidence For target page identification, we follow a language modeling approach. In Wikipedia, the title of a page is the main concept of the page and is usually the same as or similar to the anchor text linked to it. Therefore, a straightforward way of modeling this relationship would be to assume that a given anchor text can be generated by the language models that generate the titles. Thus the problem boils down to estimating the probability that the given anchor text A is generated from a given title model θ_t by applying the Bayes' Theorem.

$$P(\theta_t|A) = \frac{P(A|\theta_t)P(\theta_t)}{P(A)}, \tag{14}$$

where the $P(\theta_t)$ is assumed to be uniformly distributed and $P(A)$ is the same for each anchor and is usually dropped since it does not affect ranking. For estimating the probability $P(A|\theta_t)$, we assume each term in the anchor text to be independent and estimate the maximum likelihood (ML) of the anchor term w generated by the title model:

$$P(A|\theta_t) = \prod_{w \in A} P(w|\theta_t) \tag{15}$$

To avoid zero probabilities, we smooth $P(w|t)$ with the background collection of all page titles C_T using the Jelinek-Mercer method to obtain $P(w|\theta_t)$:

$$P(w|\theta_t) = (1 - \lambda) \cdot P(w|t) + \lambda \cdot P(w|C_T), \tag{16}$$

where

$$P(w|\theta_t) = \frac{n(w,t)}{\sum_{w'} n(w',t)} \tag{17}$$

In this equation, $n(w,t)$ and $n(w',t)$ are the number of times terms w and w' occur in a candidate target page's title t.

Target page identification - Topic Page Content Evidence Since most Wikipedia pages are short, it is very likely to end up with equal probabilities for different target candidates, especially in cases where disambiguate pages are involved. In order to solve this disambiguation problem, we try to incorporate an additional evidence source, the topic page content evidence. The assumption behind this is that if a candidate target page is the real target page for the given topic page, the content of the topic page should somehow relate to the terms in the title field of the candidate target page. Based on this assumption, we model the problem as the probability that the language model of the topic page θ_d generates the title of the candidate target page t. Similarly as before, we apply Bayes' Theorem to estimate the probability that a specific model θ_d is selected.

$$P(\theta_d|t) = \frac{P(t|\theta_d)P(\theta_d)}{P(t)}, \tag{18}$$

where $P(t)$ is ignored for ranking, and $P(\theta_d)$ is assumed to be uniformly distributed. The same ML estimate is applied to calculate $P(t|\theta_d)$:

$$P(t|\theta_d) = \prod_{w \in t} P(w|\theta_d) \tag{19}$$

Again, JM smoothing is applied to avoid zero probabilities.

3.2 Incoming Links

Our approach for identifying incoming links is quite straightforward. We experiment with two types of methods: The first method is based on exact matching of titles. We get the pages that contain the exact matches of the title of the topic page and select randomly 250 pages as linked pages. In this case, we simply ignore the context of the matched terms and assume that the existence of a link is context-independent. The second method is a rank-based approach. In this case, we perform a retrieval run on the candidate source pages, and use the title of the topic page as query. The top 250 pages are selected as the linked pages. Further thresholding on these 250 pages is also explored, and we describe this later in detail in our submitted runs section.

3.3 The LTW Runs

We submitted three runs to the LTW track. Each of the runs tries to explore the different research questions.

ltw01 : We use this run as our baseline run.
Outgoing links: We select word sequences as anchor texts whose ALR is larger than 1; use the anchor text as query on the title field of the Wikipedia collection and retrieve target pages; use the top-ranked page as the target page.
Incoming links: We use the title of the topic page as query and retrieve the top 250 pages from the collection as the source pages that are linked to the given topic.

ltw02 :We compare this run with ltw01. For outgoing links, we use this run to test if re-ranking would help disambiguate the target pages. For incoming links, this run tests how much the term linked with a topic page is related with the content of the source page.
Outgoing links: We select anchors with ALR larger than 1; retrieve the target page whose title matches (exact or partial) the anchor text; re-rank the target pages according to the likelihood of the target title in the topic page (i.e., Eq. 18), and select the top-ranked page as the target page.
Incoming link: use the topic title to find exact matches in the collection, select randomly 250 pages as the incoming source page.

ltw03 : This run checks an assumption: the linked pages should be semantically close if the links are not generated automatically by Wikipedia (i.e., country, year, etc.). This run does not try to identify anchor text at all.
Outgoing links: We use the topic title as query to retrieve 250 candidate target pages, rank them by cosine similarity to the topic page, and select the target pages whose similarity is larger than 0.15.
Incoming links: We use the same strategy as that of the outgoing links, but select the source pages whose similarity is larger than 0.026.

Here, 0.15 is the average cosine similarity between linked pages that are sampled from the collection; 0.026 is the threshold for "exceptionally" high similarity between two random pages. Different thresholds mean we would like outgoing links to emphasize on precision, and incoming links to emphasize on recall.

3.4 Experimental Settings and Results

In this section, we discuss the experimental settings in generating our runs and present the evaluation results, followed by a discussion of the initial observations.

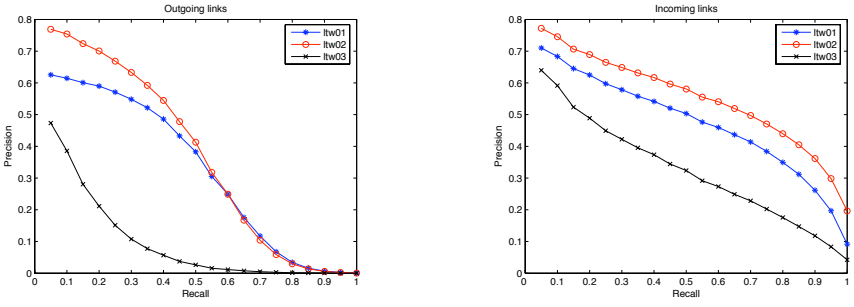
We use the content only Wikipedia pages as input. All the XML-tags are removed and the XML-structures of the documents are ignored in our experiments. For preprocessing, we use a Porter stemmer for both topic pages and target pages without stop words removal. All the topic pages are virtually deleted from the collection. For all retrieval experiments, we use the Lemur toolkit [1]. Smoothing parameter λ (Eq. 16) is heuristically set to 0.1 for all experiments (i.e. the background statistics have little impact).

Table 1 lists the evaluation results of the submitted runs in terms of MAP. Figure 1 shows the precision-recall plots for both incoming and outgoing links. In terms of MAP, we see that for both outgoing links and incoming links, run *ltw02* has better performance than *ltw01*. For outgoing links, the re-ranking with content evidence does help improve the performance, especially in early precision. This shows in 1(a). For incoming links, the fact that *ltw02* is better than *ltw01* suggests that simple title match works better than a rank-based method. Run *ltw03* performs consistently worse than the other two runs. However, early precision for this run is not that bad, suggesting that the similarity between pages could be a reasonable feature.

[1] http://www.lemurproject.org

Runs	MAP of outgoing links	MAP of incoming links
ltw01	0.2879	0.4800
ltw02	**0.3474**	**0.5249**
ltw03	0.1041	0.3345

Table 1. Results of submitted runs



(a) Precision-Recall plot for outgoing links (b) Precision-Recall plot for incoming links

Fig. 1. Precision-Recall plots for the submitted runs

4 Conclusions

We described our approaches, submissions, and initial results of this year's INEX participation. For the Entity Ranking track our focus was on the entity ranking and list completion tasks, and our chief aim was to develop a general language modeling framework to model these. Given the models we developed, we are left with plenty of choices on how to estimate the various components these models offer. For most of the components we applied simple options, that mainly make use of the category information that is available in Wikipedia. More elaborate ways of estimating the components of our models are left to future work and depend on the results of this year's participation. Results and conclusions are postponed in this paper due to the lack of evaluation results at the time of writing.

As to the Link-the-Twiki track, we submitted three runs for the File-to-File task designed to examine different features for file-level link generation. For outgoing links we based our runs on a two-step procedure: anchor text detection and target page identification. For anchor text detection, we use the anchor likelihood ratio (ALR), and for target identification, we apply a language modeling approach with different sources of evidence (i.e. title field and topic page content). Results show that the title field evidence alone is an important feature, and early precision can be improved by adding content evidence. For incoming links, we apply two very simple methods: exact matching and retrieval using the title of the topic page. We also submitted a run that tests if simple similarity between pages is sufficient as an indication of a link. The result shows that this is a reasonable feature, but on its own it is not powerful enough for link detection. Future work focuses on exploring more robust ways to model these features.

References

1. Balog, K., Weerkamp, W., de Rijke, M.: A few examples go a long way: constructing query models from elaborate query formulations. In: SIGIR '08. (2008)
2. Mackay, D.J.C., Peto, L.: A hierarchical dirichlet language model. Natural Language Engineering **1**(3) (1994) 1–19
3. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: CIKM '07: Proceedings of the sixteenth ACM conference on information and knowledge management, New York, NY, USA, Acm (2007) 233–242