A Generative Language Modeling Approach for Ranking Entities

Wouter Weerkamp, Krisztian Balog, and Edgar Meij

University of Amsterdam, ISLA, Science Park 107, 1098XG Amsterdam, The Netherlands,
{w.weerkamp,k.balog,edgar.meij}@uva.nl

Abstract. We describe our participation in the INEX 2008 Entity Ranking track. We develop a generative language modeling approach for the entity ranking and list completion tasks. Our framework comprises the following components: (i) entity and (ii) query language models, (iii) entity prior, (iv) the probability of an entity for a given category, and (v) the probability of an entity given another entity. We explore various ways of estimating these components, and report on our results. We find that improving the estimation of these components has very positive effects on performance, yet, there is room for further improvements.

1 Introduction

The Enitity Ranking track of this year's INEX features three tasks: *entity ranking*, *list completion*, and *entity relation search* [3]. 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 [4], and an entity is a Wikipedia article by definition.

The entity ranking task aims at retrieving entities given a certain topic and Wikipedia category: the goal is to identify the entities that are relevant given the topic and fit within the given category. The list completion task is slightly different and aims at adding entities of the same type to a small sample set of entities. Again, we also have the topic and category available, but as additional information we get one or more entities as examples.

In our participation we use a generative language modeling approach to model both tasks. This approach has been successfully applied in many information retrieval tasks [1, 5, 7, 8, 10]. Language models are attractive because of their foundations in statistical theory, the great deal of complementary work on language modeling in speech recognition and natural language processing, and the fact that very simple language modeling retrieval methods have performed quite well empirically.

A large portion of this paper is directed to the modeling of entity ranking and the estimation of the various components this framework offers us. We submitted a total of six runs, again with a focus on the entity ranking task (four runs).

The remainder of this paper introduces the modeling of the entity ranking task in Section 2, and of the list completion task in Section 3. Next, we discuss the estimation of the various components of both models in Section 4 and the experimental setup

in Section 5. The submitted runs (Section 6) and their results and discussion of these results (Section 7) follow, and we conclude in Section 8.

2 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 and, moreover, 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 and drop the denominator $P(q)$ which is 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.

3 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. 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.

4 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 $P(e|e')$.

4.1 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 5.

4.2 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.

Balog et al. [2] introduce various methods 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) employing 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 [2] to estimate the expanded query model $P(t|\hat{q})$.

4.3 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:

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

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 Wikipedia categories, by expanding the set of target categories with their subcategories, up to certain depth. Based on preliminary experiments, we set the maximum depth to three.

4.4 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.

5 Experimental Setup

5.1 Document Representation

Besides representing the entity (Wikipedia page) by its entire textual content (referred 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 the sake of comparability of runs, we use the full representation in almost all cases.

5.2 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*).

5.3 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 (i.e., the number of terms in the entity's representation). Essentially, the amount of smoothing is proportional to the length of the entity (and is like Bayes smoothing with a Dirichlet prior [6]). If there is very few content available for the entity (i.e., the corresponding article is very short) 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.

5.4 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.

6 Submitted Runs

This section lists our submitted runs (six in total) and the configuration used for each (note that all runs use the full representation, unless stated otherwise). For the entity ranking task the following four runs were submitted:

6_UAms_ER_T_baseline: Our baseline run using Eq. 3.
3_UAms_ER_TC_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_TC_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_TC_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 were submitted for the list completion task:

5_UAms_LC_TE_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_TCE_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.

7 Results and Discussion

We report on the results of our submissions and the best performing run among all submitted runs for each task. The metric used for measuring performance is *xinfAP* [9]. Significance is tested against our best performing run, using a two-tailed paired t-test and $\alpha = .05$.

Run	xinfAP
1_FMIT_ER_TC_nopred-cat-baseline-a1-b8	0.341
3_UAms_ER_TC_overlap	0.253
4_UAms_ER_TC_cat-exp	0.232
1_UAms_ER_TC_mixture	0.222 *
6_UAms_ER_T_baseline	0.111

Table 1. Results for the entity ranking task. Significant differences against our best performing run are marked with *.

Table 1 presents the results for the entity ranking task, in decreasing order of performance. Our main finding are as follows. Taking category information into account (*6_UAms_ER_T_baseline* vs. the other runs) improves performance. Despite the apparent increase over the baseline, in terms of xinfAP scores, these differences are not significant. We leave the investigation of this to further work. Adding additional features (besides using category information) seems less beneficial; query expansion using the category feedback method (method (ii) in Section 4.2) has a slight negative effect on performance (*4_UAms_ER_TC_cat-exp* vs. *3_UAms_ER_TC_overlap*). Mixing the two document representations, *full* and *paragraph*, hurts performance (*1_UAms_ER_TC_mixture* vs. *3_UAms_ER_TC_overlap*), in this case significantly so.

Run	xinfAP
1_FMIT_LC_TE_nopred-stat-cat-a1-b8	0.402
2_UAms_LC_TCE_dice	0.319
5_UAms_LC_TE_baseline	0.133 *

Table 2. Results for the list completion task. Significant differences against our best performing run are marked with *.

Results for the list completion task are shown in Table 2. We observe that adding estimates of $P(e|e')$ and $P(e|C)$ to the baseline yields in substantial and significant improvements over our (naive) baseline. Further analysis is needed to see which of the components makes up for most of this improvement.

8 Conclusion

We described the approach, submitted runs, and results of our participation in this year's INEX Entity Ranking track. Our focus lied on the entity ranking and list completion tasks, and our chief aim was to develop a general language modeling framework to model these in a uniform and theoretically sound way. Given the models we introduced, we are left with plenty of choices on how to estimate the various components these models offer. For most of these components we applied simple options which mainly make use of the category information available in Wikipedia. Results show that using category and entity information in different ways leads to increases in performance over simple baselines, yet there is room for further improvements. Based on these initial findings, in further work we aim at investigating additional, more elaborate ways of estimating the various components, and exploring their impact on performance. Since we made quite strong independence assumptions regarding our input variables, in another line of work we are planning on examining these dependencies.

Bibliography

[1] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR '06*, pages 43–50, 2006.

[2] K. Balog, W. Weerkamp, and M. de Rijke. A few examples go a long way: constructing query models from elaborate query. In *SIGIR '08*, pages 371–378, 2008.

[3] G. Demartini, A. P. de Vries, T. Iofciu, and J. Zhu. Overview of the INEX 2008 entity ranking track. In *This volume*, 2009.

[4] L. Denoyer and P. Gallinari. The Wikipedia XML corpus. *SIGIR Forum*, 40(1): 64–69, 2006.

[5] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, University of Twente, 2001.

[6] D. J. C. Mackay and L. Peto. A hierarchical dirichlet language model. *Natural Language Engineering*, 1(3):1–19, 1994.

[7] D. Miller, T. Leek, and R. Schwartz. A hidden Markov model information retrieval system. In *SIGIR '99*, pages 214–221, 1999.

[8] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98*, pages 275–281, 1998.

[9] E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating ap and ndcg. In *SIGIR '08*, pages 603–610, 2008.

[10] C. Zhai. Statistical language models for information retrieval a critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, 2008.