Probabilistic Field Mapping for Product Search

Aman Berhane Ghirmatsion and Krisztian Balog

University of Stavanger, Stavanger, Norway
ab.ghirmatsion@stud.uis.no, krisztian.balog@uis.no,

Abstract. This paper describes our participation in the product search task of the CLEF 2015 LL4IR Lab. Working within a generative language modeling framework, we represent products as semi-structured documents. Our focus is on establishing a probabilistic mapping from query terms to document fields. We present and experimentally compare three alternatives. Our results show that term-specific mapping is beneficial. We also find evidence suggesting that estimating field mapping priors based on historical clicks outperforms the setting where the priors are uniformly distributed.

1 Introduction

Online shopping has become a common practice and one of the most popular activities people perform on the Internet. Its success can be credited to ease and convenience, large selection of products, 24/7-availability, low pricing, just to mention some of the benefits. Just with any other website or online service, effective information retrieval systems are indispensable for e-commerce sites. Webshops need to offer search functionality that makes their available products easy to find. Providing customers with a positive shopping experience can increase sales, and is thereby essential to success of the business. Users are accustomed to the "single search box" paradigm and expect the search engine to understand their information need. In this paper we address the task of *ad-hoc product search*, defined as follows: given a keyword query, return a ranked list of products from a product catalog that are relevant to the query.

Products are described by a number of attributes, such as name, brand, description, categories, and so on. These attributes have a specific semantic meaning (one that can even be known a priori), albeit the amount of text associated with each is typically rather small. Recognizing for each query term which of the product attributes (if any) is targeted, therefore, is believed to lead to improved retrieval performance. Our objective in this work is to develop, implement, and evaluate methods that establish a mapping from individual query terms to specific product fields. We represent products as semi-structured documents (following the predominant approach to entity retrieval [1]) and employ the Probabilistic Retrieval Model for Semistructured Data (PRMS) [2], a model that is known to perform well under conditions where the collection is homogeneous and fields have distinctive term distributions [1]. Our setting is exactly like that. Given the inherent uncertainty, the term-field mapping is represented as probability distribution over fields. We decompose the estimation into term-field probability and prior field probability components and propose three specific instantiations of the PRMS model. We evaluate our approaches and report results using the LL4IR platform.

Table 1. Fields in our index.

Field	Description
`brand`	Product's brand
`category`	Leaf level product category
`characters`	Characters associated with the product (e.g., Barbie)
`contents`	Catch-all contents field
`description`	Full textual description
`product_name`	Product's name
`queries`	Queries that led to the product
`main_category`	Top level product category
`short_description`	Short textual description

2 Task and Data

We address the task of ranking products (from a product catalog) in response to a (short) keyword query. In this section we resort to a brief description of what and how we used from the living labs API in our participation. For a detailed description of the task and setup, we refer to the LL4IR Lab overview paper [5].

For each query, the living labs platform makes a set of candidate products available via the `doclist` API endpoint. For each of these products, the details can be requested via the `doc` API endpoint, which provides a product description in the form of key-value pairs. We built a single Lucene index from all unique products made available to us (i.e., products from all doclists) with the fields shown in Table 1. Note that `contents` is a catch-all field that does not come from the API; it is the concatenation of all field content associated with the product, created at indexing time. In addition, we also used the `historical` API endpoint to obtain aggregated click-through rate (CTR) for the training queries.

We produce rankings offline using the retrieval framework and field-mapping methods described in Sections 3 and 4. Once uploaded to the API, these are interleaved with and compared against the site's production ranking system. The numbers reported in Section 5 are obtained via the `outcome` API endpoint.

3 Retrieval Framework

We base our approach on a generative language modeling framework. Language models provide a transparent and effective means for incorporating structural cues. They are especially appropriate under circumstances where training data is scarce. Our goal with this section is merely to present a brief introduction to the general framework and notation we use, thereby making the paper self-contained; for a more detailed description of these models we refer the reader to [1].

3.1 Standard Language Modeling Approach

The standard language modeling approach works as follows. Given a query q and a document d, the relevance of the document with regard to the query can be expressed

as the conditional probability $P(d|q)$. Using Bayes theorem, $P(d|q)$ is rewritten as shown in Eq. 1 below. Since the $P(q)$ is identical for all candidate documents, it can be safely discarded. Thus, the posterior probability $P(d|q)$ is given by the product of query generation probability $P(q|d)$ and document prior probability $P(d)$:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \approx P(q|d)P(d). \tag{1}$$

Assuming uniform document priors, the ranking of documents is then proportional to $P(q|d)$. The simplest solution to estimating this probability is by assuming a bag-of-words document representation. We let θ_d be an unigram document language model where $P(t|\theta_d)$ expresses the probability of term t given the document. Ranking documents is done according to the probability that a query q is observed during repeated random sampling from the model of document d (where $n(t, q)$ is the number of times t occurs in q):

$$P(q|\theta_d) = \prod_{t \in q} P(t|\theta_d)^{n(t,q)}. \tag{2}$$

To estimate $P(t|\theta_d)$ we use a linear combination of maximum-likelihood document and collection language models (i.e., employ Jelinek-Mercer smoothing):

$$P(t|\theta_d) = (1 - \lambda)P_{ML}(t|d) + \lambda P_{ML}(t|C), \tag{3}$$

where $P(t|d)$ and $P(t|C)$ are relative frequencies of term t in the document and in the collection, respectively, and λ is the smoothing parameter.

3.2 Mixture of Language Models

The standard language modeling approach treats the document as "flat text," and is not able to make use of various document fields. An extension called Mixture of Language Models (MLM) is proposed by Ogilvie and Callan [3], where a separate language model is calculated for each document field, and these field language models are then combined into a single document-level representation. Under this approach the document language model is taken to be:

$$P(t|\theta_d) = \sum_{f \in F} \alpha_f P(t|\theta_{d_f}), \tag{4}$$

where f is a field from the set of available fields F, α_f is the relative importance of the field such that $\sum_{f \in F} \alpha_f = 1$, and θ_{d_f} is a field-specific language model. The field language model can be estimated the same way as the document language model, except that term occurrences are considered only within the given field (this is indicated by the f subscript):

$$P(t|\theta_{d_f}) = (1 - \lambda_f)P_{ML}(t|d_f) + \lambda_f P_{ML}(t|C_f). \tag{5}$$

Ogilvie and Callan [3] suggest to set the field weights proportional to the length of the fields or to their individual retrieval performance. We also use the latter approach in one of our methods. We set the smoothing parameter to be the same for all fields: $\lambda_f = 0.1$.

Table 2. Overview of field mapping methods.

Method	$P(t\|f)$	$P(f)$
Method 1	$\frac{1}{\|F\|}$	$\propto \mathrm{NDCG}_f$
Method 2	$P_{ML}(t\|C_f)$	$\frac{1}{\|F\|}$
Method 3	$P_{ML}(t\|C_f)$	$\propto \mathrm{NDCG}_f$

3.3 Probabilistic Retrieval Model for Semistructured Data

Instead of using a fixed field weight that is the same for all query terms, Kim et al. [2] propose the use of *mapping probabilities*. In their approach, called Probabilistic Retrieval Model for Semistructured Data (PRMS), α_f in Eq. 4 is replaced with $P(f|t)$: the probability of term t being mapped to field f. The estimation of the document language model then becomes:

$$P(t|\theta_d) = \sum_{f \in F} P(f|t) P(t|\theta_{d_f}). \tag{6}$$

For estimating the mapping probabilities we again make use of Bayes' theorem:

$$P(f|t) = \frac{P(t|f)P(f)}{P(t)} = \frac{P(t|f)P(f)}{\sum_{f' \in F} P(t|f')P(f')}, \tag{7}$$

where $P(t|f)$ can be estimated conveniently by $P_{ML}(t|C_f)$, and $P(f)$ is a prior mapping probability. We shall present multiple alternatives for setting the components of Eq. 7, $P(t|f)$ and $P(f)$, in the next section.

4 Estimating Mapping Probabilites

We present three specific instantiations of the PRMS model, shown in Table 2, that only differ in the estimation of the mapping probability, given in Eq. 7. This formula has two components that need to be defined: the term-field probability $P(t|f)$ and the field prior $P(f)$. Let us point out that MLM can be seen as a special case of PRMS, where $P(t|f) = 1/|F|$ and $P(f) = \alpha_f$.

The probability of a term occurring in a given field, $P(t|f)$, can be taken uniform (i.e., $1/|F|$) or calculated using the term counts in field f across the whole collection, $P_{ML}P(t|C_f)$, as it is done in the original PRMS approach [2].

For the field prior we consider a uniform and a non-uniform setting. In case of the latter, we capitalize on the fact the we have access to training data that can be used for evaluating the retrieval performance of each field individually. Specifically, we rank training queries based on $P(q|\theta_{d_f})$ and measure retrieval performance in terms of NDCG, averaged over all training queries (denoted as NDCG_f). The gain of a document is set to historical CTR (as provided by the historical feedback API endpoint). $P(f)$ is then set proportional to NDCG as follows:

$$P(f) = \frac{\mathrm{NDCG}_f}{\sum_{f' \in F} \mathrm{NDCG}_{f'}}. \tag{8}$$

Table 3. Field priors based on individual field performance.

Field name	NDCG_f	$P(f)$
Brand	0.0684	0.0240
Product name	0.5632	0.1989
Characters	0.3792	0.1339
Category	0.4305	0.1520
Description	0.3919	0.1384
Short description	0.2986	0.1054
Contents	0.6987	0.2468

Table 4. Offline results for the training queries. Best scores for each metric are in boldface.

Method	MAP	MRR	NDCG
Method 1	**0.8118**	**0.9948**	**0.7045**
Method 2	0.7916	**0.9948**	0.7012
Method 3	0.7997	**0.9948**	0.7026

Table 3 shows the individual performances and the estimated priors for the fields in our index. Recall that `contents` is a catch-all field, containing all content available for the given product.

5 Results

We report on two sets of experiments: traditional, test collection based *offline* results on the training queries and *online* results collected via the living labs platform on the test queries.

5.1 Training Queries

We use historical CTR as ground truth. For binary relevance (MAP and MRR) we consider each product with at least 0.001 CTR as relevant; for graded relevance (NDCG) we use CTR as the gain value. We note that for Methods 1 and 3 we train and test on the same set of queries. The numbers reported here are only meant to show how well the models can be fitted to the data.

Table 4 presents the results. We find that all three methods perform very similarly for all three metrics. While they achieve virtually perfect MRR, i.e., the first ranked results is always a relevant one, there is room for improvement in terms of MAP and NDCG. Our CTR-based ground truth is likely to be biased based on the site's existing ranking. Therefore, a comparison against the production system would reveal more about the differences between our methods; this is exactly what follows next.

Table 5. Online results for the test queries. In addition to outcome, we also report on the total number of impressions (#I) and percentage of wins (%W), losses (%L), and ties (%T). Best scores for each metric are in boldface.

Method	Run ID	Round #1					Round #2				
		Outcome	#I	%W	%L	%T	Outcome	#I	%W	%L	%T
Method 1	`UiS`	0.2827	699	7.7	19.6	72.7	0.4118	731	**11.5**	16.4	72.1
Method 2	`Mira`	**0.3413**	725	**9.8**	18.9	71.3	0.4389	757	10.4	13.3	76.2
Method 3	`Jern`	0.3277	665	8.7	**17.9**	73.4	**0.4795**	767	10.7	**11.6**	77.7

5.2 Test Queries

We submitted three rankings corresponding the Methods 1–3, as shown in Table 5. The main evaluation metric is *outcome*, which is defined as:

$$\text{outcome} = \frac{\#\text{wins}}{\#\text{wins} + \#\text{losses}}, \tag{9}$$

where the number of wins and losses are measured against the site's production ranking system. Additionally, we report on the total number of impressions and the ratio of wins, losses, and ties.

Before discussing our observations, it is important to mention that the way interleaving with the production system is performed has changed from Round #1 to Round #2, to deal correctly with unavailable products (as explained in [4]). It is therefore believed that the Round #2 results are more reliable. Nevertheless, there are some open questions, such as how many impressions are needed before one can draw firm conclusions, and whether the results obtained in Round #2 are an accurate reflection of the performance of our methods.

It is immediately apparent that the Round #2 numbers are higher, due to the aforementioned change to interleaving. Outcomes for Round #1 are above the (corrected) expected outcome of 0.28, while for Round #2 they are below the expected outcome of 0.5. For neither round were we able to outperform the organizers' baseline (with an outcome of 0.4691 and 0.5284 for Rounds #1 and #2, respectively), which clearly shows that there is considerable room for improvement.

Concerning the comparison of Methods 1–3, we make the following observations. All methods received about the same number of impressions, the relative difference between them is within 10%. In over 70% of the cases there is a tie between the experimental and production rankings; this is the same for all three methods. It is clear that Method 1 is the worst performing out of the three; it is expected, as this method is essentially MLM, which performs the field mapping independent of the query terms. As for Method 2 vs. 3, the results are mixed. In Round #1 Method 2 performed slightly better ($+4\%$ over Method 3), while in Round #2 Method 3 came first ($+9\%$ over Method 2). We further observe that Method 3 has the most ties with the production system and it is also the one with the least number of losses against it. Based on these results we can safely conclude that term-specific mapping is beneficial (Method 1 vs. Methods 2 and 3). There is also evidence suggesting that non-uniform field priors are preferred and that historical CTR offers a simple and intuitive way of setting them (Method 2 vs. 3).

Table 6. Field mapping probabilities.

Query	Term	Field name	M1	M2	M3
baba	baba	product_name	0.1989	0.2509	0.2713
		contents	0.2468	0.2592	0.3509
		category	0.1520	0.3723	0.3024
		short_desc	0.1054	0.0658	0.0356
		desc	0.1384	0.0516	0.0391
pötyi	pötyi	product_name	0.1989	0.1067	0.0877
		contents	0.2468	0.8820	0.9060
		desc	0.1384	0.0109	0.0060
minnie	minnie	product_name	0.1989	0.0860	0.1156
		contents	0.2468	0.0860	0.1479
		characters	0.1339	0.8102	0.7214
		short_desc	0.1054	0.0117	0.0080
		desc	0.1384	0.0072	0.0068
bogyó és babóca	bogyó	product_name	0.1989	0.0610	0.0861
		contents	0.2468	0.0633	0.1118
		desc	0.1384	0.0034	0.0034
		short_desc	0.1054	0.0062	0.0044
		characters	0.1339	0.8657	0.7941
	és	product_name	0.1989	0.0874	0.1218
		contents	0.2468	0.0984	0.1714
		desc	0.1384	0.2578	0.2514
		short_desc	0.1054	0.2316	0.1613
		characters	0.1339	0.3246	0.2939
	babóca	product_name	0.1989	0.0594	0.0823
		contents	0.2468	0.0875	0.1514
		desc	0.1384	0.0034	0.0033
		short_desc	0.1054	0.0061	0.0042
		characters	0.1339	0.8434	0.7587

6 Analysis

In our analysis section we ask the following question: How different are the mappings created by the different methods? We answer this both qualitatively and quantitatively. For the former, Table 6 shows the estimated mapping probabilities for a number of training queries; we can see that these are indeed different. As for the latter, Table 7 presents the Kendall's rank correlation coefficient (τ); the numbers are averages computed over all queries (both training and test). We observe that the correlation between all three methods is high and that Methods 2 and 3 are more similar to each other than they are to Method 1. The level of similarity ($\tau = 0.95$) between Methods 2 and 3 explains why it is so difficult to decide a winner. Our findings concerning Method 2 vs. 3 (uniform vs. non-uniform field priors) should therefore be taken with a grain of salt.

Table 7. Similarity of rankings measured by the Kendall rank correlation coefficient (τ).

	Method 2	Method 3
Method 1	0.867	0.864
Method 2	–	0.950

7 Conclusions

We have described our participation in the product search task of the CLEF 2015 LL4IR Lab. Our focus has been on various field mapping approaches in a language modeling framework, which we compared experimentally. We have shown that using term-specific mapping has a positive effect on retrieval performance. We have also presented evidence suggesting that estimating field mapping priors based on historical clicks outperforms the setting where the priors are uniformly distributed. While we were able to observe interesting differences between our methods, further work is needed to be able to outperform the site's production ranking system.

Bibliography

[1] K. Balog. Semistructured data search. In N. Ferro, editor, *Bridging Between Information Retrieval and Databases*, volume 8173 of *Lecture Notes in Computer Science*, pages 74–96. Springer Berlin Heidelberg, 2014.

[2] J. Kim, X. Xue, and W. B. Croft. A probabilistic retrieval model for semistructured data. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 228–239. Springer-Verlag, 2009.

[3] P. Ogilvie and J. Callan. Combining document representations for known-item search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 143–150. ACM, 2003.

[4] A. Schuth, K. Balog, and L. Kelly. Extended overview of the living labs for information retrieval evaluation (LL4IR) CLEF lab 2015. In *CLEF 2015 Labs and Workshops, Notebook Papers*. 2015.

[5] A. Schuth, K. Balog, and L. Kelly. Overview of the living labs for information retrieval evaluation (LL4IR) CLEF Lab 2015. In *Sixth International Conference of the CLEF Association, CLEF'15*, volume 9283 of *Lecture Notes in Computer Science (LNCS)*. Springer Berlin Heidelberg, 2015.