Query and Product Suggestion for Price Comparison Search Engines Based on Query-product Click-through Bipartite Graphs

Lucia Noce, Ignazio Gallo and Alessandro Zamberletti
University of Insubria,
Department of Theoretical and Applied Science,
Via Mazzini, 5, 21100 Varese, Italy
{*lucia.noce, ignazio.gallo* }*@uninsubria.it*

Keywords: Query Suggestion, Product Suggestion, Price Comparison Engines

Abstract: Query suggestion is a technique for generating alternative queries to facilitate information seeking, and has become a needful feature that commercial search engines provide to web users. In this paper, we focus on query suggestion for price comparison search engines. In this specific domain, suggestions provided to web users need to be properly generated taking into account whether both the searched and the suggested products are still available for sale. To this end, we propose a novel approach based on a slightly variant of classical query-URL graphs: the query-product click through bipartite graph. Such graph is built using information extracted both from search engine logs and specific domain features such as categories and products popularities. Information collected from the query-product graph can be used to suggest not only related queries but also related products. The proposed model was tested on several challenging datasets, and also compared with a recent competing query suggestion approach specifically designed for price comparison engines. Our solution outperforms the competing approach, achieving higher results both in terms of relevance of the provided suggestions and coverage rates on top-8 suggestions.

1 INTRODUCTION

Query suggestion plays an important role in helping users to find what they are looking for when querying web search engines. It is a particularly interesting research topic because the usability, popularity and success of web search engines are strongly related to the effectiveness of the helping tools provided to users while performing textual queries: the larger the amount of users that issue queries leading to optimal/desirable results, the higher is the usability of the web search engine and consequently the larger is the amount of users that will keep using that web search engine for other future queries.

Although many effective and sophisticated query suggestion algorithms that have been proposed in literature are currently employed by web search engines to improve user search experience (Kato et al., 2013), precisely understanding in a limited amount of time the needs of service users from textual queries is still a challenging task that has yet to find an optimal solution. In fact, achieving satisfying query suggestion accuracies for textual queries is a non-trivial task because most user-made textual queries are typically very short and contain ambiguous words, and it is therefore difficult to provide good query suggestions without using complex pipelines.

Most query suggestion algorithms in literature focus on improving user search experience by suggesting related queries from a given textual query, trying to guide users in finding what they are looking for. The list of suggestions is usually created by mining related queries from web search engine logs and session information, trying to gather previous users knowledge. Depending on the chosen approach, query suggestion techniques can be grouped into two categories (Meng, 2014): session-based methods and click-through-based methods. The first approach uses the consecutiveness of the queries to model user behavior within each user session, while the second one focuses on the relationship between the submitted queries and the URLs clicked by the user. Both session-based methods and click-through-based methods are being used by many commercial web search engine, *e.g.* Google, Ask and Yahoo!, to provide correlated queries to improve usability.

Despite most of the query suggestion works in literature were developed for web search engines, query

suggestion is also important for e-commerce platforms and price comparison search engines (Hasan et al., 2011). In this work we present a click-through-based query suggestion system specifically designed for price comparison websites (ShoppyDoo, 2015). It provides query suggestions on the basis of all the specific product information available in this context, *e.g.* the category of the product that the user is browsing while issuing the query and the clicked product. To this end, instead of using the URLs clicked by users, our click-though based method exploits the information from clicked products.

Query suggestion in price comparison websites is an even more challenging task because, for example, it may be disadvantageous to suggest a query for which there are no available product offers.

2 RELATED WORKS

Query suggestion is an interesting research topic widely investigated in researches on Internet search domain (Kim and Croft, 2014; Song et al., 2012; Jiang et al., 2014). Algorithms for query suggestion can be classified as either session-based (Zhang and Nasraoui, 2008; Boldi et al., 2009; Lau and Horvitz, 1999; Zanon et al., 2012; Ozmutlu and Spink, 2003) or click-through-based (Mei et al., 2008; Ma et al., 2010; Song and He, 2010).

Session-based methods assume that users submit queries in a *sequential manner*. The basic assumption is that when a badly written textual query is executed by a user, there will always be a correct version of the wrong textual query previously executed by the same user that follows the wrong one. The sequence of queries is usually obtained by exploiting user sessions, and it is used to extract the user's intentions. Among session-based query suggestion methods, the works that are closely related to our are the ones of (Lau and Horvitz, 1999; Ozmutlu and Spink, 2003; Boldi et al., 2009; Zanon et al., 2012).

In their work about query refinement Lau *et al.* (Lau and Horvitz, 1999) assert that, after a failed search, most users refine their original query either by adding more details or by executing a new one. Another study by Ozmutlu and Spink (Ozmutlu and Spink, 2003) demonstrates that in 88.6% of cases a query session relates to a just a single topic, supporting the thesis that query sessions are useful to extract information that can be used for query suggestion purposes.

Following this approach, Boldi *et al.* (Boldi et al., 2009) propose a system that uses query-flow graphs built from query session logs. In the query-flow graph, there exists an edge from a query q_1 to a query q_2 if and only if both queries belong to the same session. Edges and vertexes can be weighted. These weights represent the frequency of a query and the popularity of a transition for a vertex and for an edge respectively. Edges are also labelled with some extra information describing the nature of the transition (*e.g.* specialization, probability that user moves from q_1 to q_2, *etc.*). In their experimental evaluation, Boldi *et al.* show that adding weights and labels to query-flow graphs allows the creation of a session-based query suggestion algorithm which achieves the same results of click-through-based models.

A session-based query suggestion method specifically designed for price comparison engines has been proposed by Zanon *et al.* (Zanon et al., 2012). The model performs query suggestion system using the same query-flow graphs proposed by Boldi *et al.* (Boldi et al., 2009). Zanon *et al.* adapt the model of Boldi *et al.* to their context exploiting the fact that in price comparison engines offers are grouped into categories. Their model reach a quality of 70% and a coverage of 37% for top-8 suggestions.

Click-through-based methods exploit the information of the URL that a user clicks after having submitted a query. These category of approaches assume that two queries are similar if they share a consistent number of clicked URLs.

Baeza-Yates *et al.* (Baeza-Yates et al., 2004) exploit click-through data in their query recommendation method. The presented methodology is based on a clustering process over data extracted from the query log, where suggested queries are proposed according to a rank score evaluated in terms of similarity and support of the queries.

Following Baeza-Yates *et al.* work, a query suggestion method for e-commerce was proposed by Hasan *et al.* (Hasan et al., 2011). Authors discuss about the challenges related to their context and underline that, due to the nature of the data belonging to e-commerce websites, it could be difficult to adapt session-based query suggestion approaches to e-commerce systems. Instead, a common approach in this field is to use graph representation for underlining queries and URLs relationship through a click-through bipartite graph. Click-through bipartite, represents an implicit judgment given by users of the relationships between queries. It has been proven that this type of graph is a very precious source of information for measuring similarities among its components (Wu et al., 2013).

Many click-through-based approaches exploit bipartite graph, *e.g.* Ma *et al.* (Ma et al., 2010) proposed a ranking method that is able to diversify query sug-

gestion results, employing Markov random walk process and hitting time analysis; Song *et al.* use bipartite graph to analyze both clicked URLs and skipped URLs to recommend related queries (Song and He, 2010); Cao *et al.* (Cao et al., 2008) exploit click-through data and group similar queries into *concepts*, providing query suggestions based on these *concepts*.

3 PROPOSED METHOD

Query suggestion in price comparison websites is very similar to the query suggestion in e-commerce websites, because in both scenarios users search for products they want to buy and therefore it is reasonable to say that they issue to the two different search engine the same type of queries. Following the discussion of Hasan *et al.* (Hasan et al., 2011) about e-commerce websites, the proposed method is a click-through-based query suggestion approach.

Our model exploits the two main entities of the website for which it has been designed for (Shoppy-Doo, 2015): products and categories. These concepts are not restricted to our context but are used in many other major e-commerce and search engine websites.

More in details, the result of a generic query in our context is a list of commercial products sold by several retailers and sorted by descending price. Each item of the list is an offer. All the offers belong to a specific category, according to the type of the objects searched by the user. Offers may also be linked to products. Products represent aggregations of items/offers, and all the offers inherent to a specific item are collected under the related product.

In general, a click-through-based query suggestion algorithm is composed of two phases: (i) an offline module which manages the data and builds the model, (ii) and an online procedure which supplies the query suggestions. Our solution extracts information from web server logs. Starting from those logs, the offline phase of our method can be summarized in the following steps:

- Data Extraction: is the pre-processing phase, the web server log is parsed. Queries and related information are extracted, cleaned and normalized;
- Session creation: the sessions are created as associations between queries and clicked products;
- Building phase: creation of the click-through bipartite graph by exploiting the notion of product;

On the other hand, the subsequent online phase gathers query suggestions from the model and returns a fixed number of ranked related queries and products.

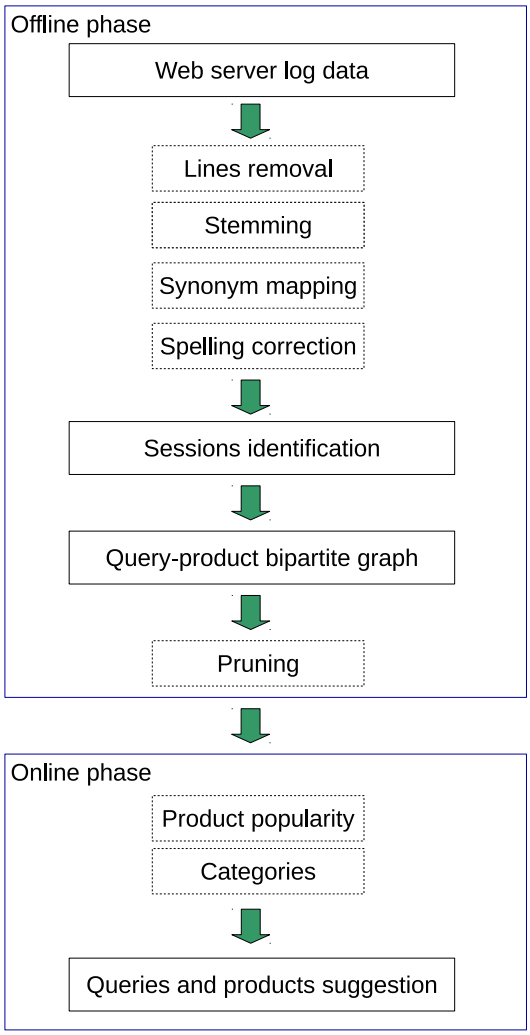

Figure 1: Visual summarization of the proposed pipeline. From top to bottom: starting from raw log data, all the information needed to determine query sessions are extracted; the query-product bipartite graph is built and suggestions are provided by exploiting both the product categories and the product popularities extra-fields.

In Figure 1, both the offline and the online phases of the proposed approach are visually summarized. In the following sections all their major details are reported.

3.1 Data Extraction

The first step consists of the extraction of data from the web server log. Those logs contains rough data that has to be cleaned before being used for query suggestion.

Different cleaning stages are performed to gather only log lines containing the information we want to exploit. First we performed bot filtering to retrieve

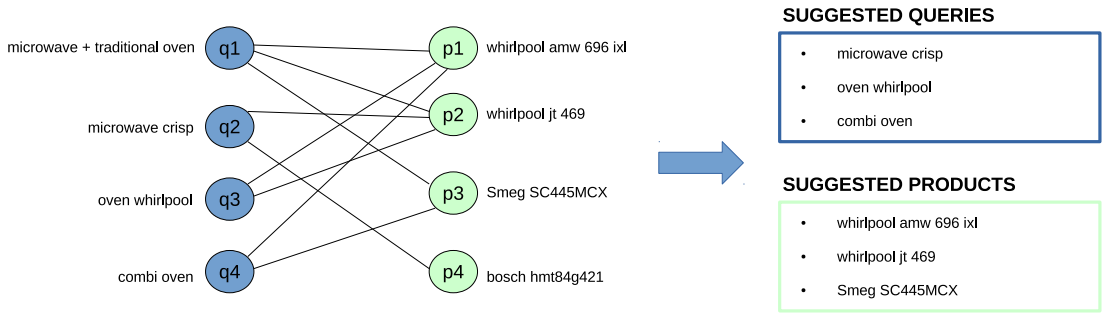

Figure 2: This visual example shows the usage of the query-product bipartite graph. Some details, such as the popularity score and the number of clicks, are omitted to make the image more readable. Assuming that the input query is equal to q_1 and that N (number of suggestions) is equal to 3, we determine p_1, p_2 and p_3 as the 3 most popular products related to the input query q_1, and we gather from them the three most clicked related queries q_2, q_3 and q_4.

only log search lines that lead to a user click. Within a log line we extract only the fields of interest: the textual query, the id of the category for which the query has been performed, and the clicked product. Synonym mapping for queries like "mtb" and "mountain bike" and spelling correction for cases such as "dylandog" corrects into "dylan dog" are also performed. We also use stemming to detect equivalent queries, *e.g.* "woman bike" is equivalent to "women bikes", "samsung galaxy S6" represents the same query as "S6 samsung galaxy". Since the website is in italian, all the reported examples have been translated for better understanding.

3.2 Session creation

For building the model, we take into account only textual queries extracted from web server logs. A query is the list of terms that a user types to search for the item he needs, the same query may be issued to the search engine several times, each submission creates a query session.

We followed the approach that has been proposed by Wen *et al.* (Wen et al., 2001) and used by Baeza-Yates *et al.* (Baeza-Yates et al., 2004) which considers a simple notion of query session that is composed of a query and the URLs clicked within its query results. In our method instead of URLs we collected the products returned by the query, as follows:

QuerySession = (searchedText, clickedProducts)

The reason why we adopt a different notion of query session lies our application context. In price comparison engines, each time a query is submitted, a list of related offers is shown. However, each offer may be available for a limited amount of time, therefore it would be senseless to gather the direct offers URLs which may have been already destroyed. Creating query suggestions dealing to unavailable offers is improper and unattractive for users, and may cause a loss of clients for the website.

The taxonomy of our price comparison website allows us to exploit the notion of product. Not every offer is related to a product, but statistical studies made by the website administrator shows that the majority (97%) of the offers dealing to a user click are related to a product. Starting from this assertion we consider only offers related to products and sessions storing products related to the submitted queries. This not only allows us to provide a more reliable query suggestion system, but also permits to supply product suggestions to users.

3.3 Building phase

We build the proposed model using the notion of query session previously described. This way we create a different version of the well-known query-URL bipartite graph in which, instead of URLs, we use clicked products.

A query-product bipartite graph consists of two disjoint sets of nodes corresponding to queries and products respectively. Intuitively, in a click-through graph vertexes on one side correspond to queries, while vertexes on the other side represents products. An edge connecting a query q to a product p exists whenever p was clicked starting from q.

A visual example of such query-product bipartite graph is given in Figure 2. The set of nodes in the left part are queries and the ones in the right part are products. An edge between a query q and an URL u indicates a user click of u when issuing q (for simplicity click numbers are omitted from the graph).

The query-product bipartite graph gathers a large amount of potential information that can be used for

query suggestion, query clustering, query reformulation and so on.

In our method we consider two kind of graphs depending on whether a user uses a category when searching for a product or not. In our price comparison website, users are allowed to submit queries within a specified category, and they expect to receive product results from that same category. Showing query suggestions outside the selected category may be annoying for the users; as such so build two different kind of graphs. The first one does not exploit the concept of category and therefore it can be used to provide query suggestion results from all the website categories. On the other hand, query suggestion results from the second graph are restricted to a specific category selected by the user. By query suggesion results we intend both queries and products.

A final pruning phase is necessary to avoid the suggestion of queries that may have a negative impact on the user experience. We start removing products that are only related to one query, and then we remove queries that are only connected to one product.

Both the two versions of the query-product bipartite graph were tested and evaluated both in terms of time needed to complete their building phase, and in term of accuracy for the provided query suggestion results. All the experiments and results are reported and discussed in Section 4.

3.4 Online query suggestion

The online phase of our method exploits two basic concepts related to the taxonomy of the price comparison engine we used in this work: categories and popularity. As previously described, for query suggestion, we take into account only offers that belong to a product.

For each product the website administrator has provided us a numeric value which indicates the popularity of an offer. This value is computed considering both the whole server traffic and the available offers in a certain period of time, and it is periodically updated.

Given a user query q, if it belongs to the query set of the bipartite graph, we automatically obtain the list of the related queries and of the related products. The retrieved product list is then sorted by ascending popularity value, and the top k products are collected. Starting from this list of top k products, another list of related queries is produced and sorted by the number of clicks stored in the product-query bipartite graph edges. The top k retrieved queries are selected and provided as results.

On the other hand if a query q does not belong to the query-product bipartite graph we use the similarity measure described by Zanon *et al.* (Zanon et al., 2012) to retrieve the more similar query q' among the bipartite graph. This measure exploits the concept of categories and the Jaccard similarity coefficient proposed by Tan *et al.* (Tan et al., 2005). Once q' is found the system acts as previously described.

Table 2: Building phase details. For each query log size (1, 2 or 3 months), the first line represents contains the size of the pruned bipartite-graph, while the second line reports the initial numbers of queries and products. Bold values denoted the final model used during the experimental phase.

# months	Time (s)	# queries	# products
3	**37.56**	**79025**	**283188**
		156434	283876
2	18.25	49855	141765
		98654	205465
1	6.61	48926	48930
		675341	62567

4 EXPERIMENTS

We conducted several experiments to measure the effectiveness of our proposed query and product suggestion method. Both query and product suggestions were evaluated with and without using category information.

To build our query-product bipartite graph we have used three months of ShooppyDoo user query logs. More in details, we have collected a total of 1147990 queries, 1096112 clicked products and we have built a final version graph containing 79025 unique queries, 283188 unique products. The total amount of available products is 295883, so with three months of data log we almost covered all the products.

In Table 2 some more details for the graph building phase are provided. We report both the time needed by our model to complete the graph building phase using one, two and three months of query logs; and the details of the product-query bipartite graph in terms of number of queries and products after and before the pruning phase. All the experiments have been performed on a Intel Core i5 @ 3.4 GHz with 12 GB of RAM; the software has been developed using Java for the pre-processing phase, and MATLAB for the building and experimental phases.

4.1 Dataset

We create a test dataset following the manner adopted by Song *et al.* (Song and He, 2010; Song et al., 2012).

Table 1: The five different categories adopted during the experimental phase have substantially different characteristics: the "Food Supplements" category is the smallest one and less products are associated to it; the "Fridge category" is the most popular one. The cardinality of each category deeply affects the quality of the query/product suggestions provided by our model, as reported in Table 3 and 4 .

Category	Tot # products	# products with offers	# offers
Mobile phone	5040	847	8110
Oven	4735	1293	13328
Fridge	8166	1908	22503
Food supplements	71	68	485
Tires	1896	1635	12900

Table 3: Query suggestion coverage rates calculated on all the 7 test dataset. The model provides better suggestions for the more populated categories (e.g. Oven) than it does for the less populated one (e.g. Tires). Also, the exploitation of the category popularity value enables to reach higher results within the "categorized" dataset compared to the "uncategorized" one.

Dataset	3 sugg.	5 sugg.	8 sugg.
Uncategorized	73.36 %	63.48 %	56.40 %
Categorized	**79.53** %	**78.94** %	**75.53** %
Mobile phone	95.21 %	92.11 %	85.32 %
Oven	92.45 %	84.90 %	81.12 %
Fridge	94.17 %	90.07 %	89.75 %
Food supplements	69.09 %	62.04 %	34.61 %
Tires	32.56 %	23.11 %	11.56 %

Table 4: Product suggestion coverage rates. The suggested products reach an high coverage rate among all the different test datasets. The lowest values are achieved applying the model to the less populated category, and the category values exploited on the "categorized" dataset helps the model in reaching the highest results.

Dataset	3 sugg.	5 sugg.	8 sugg.
Uncategorized	73,68 %	63,15%	44.68 %
Categorized	**89,42**%	**85,62**%	**81,52**%
Mobile phone	72,10%	63,22%	54,78 %
Oven	79,24%	75,47 %	70,58 %
Fridge	83,13%	78,01%	76,31 %
Food supplements	68,03 %	51,89 %	38,46 %
Tires	50,07 %	27,34 %	8,45 %

Three sets of 500 queries each are randomly selected from the query log file. The first contains 500 queries not related to any category, the second contains 500 queries related to one of the 569 categories of the websites (suggestions also belong to the same category), while the third contains a balanced equal number of queries related to 5 selected categories. For each sampled query our algorithm generates top-8 suggestions for both queries and products.

Three judges were asked to evaluate the quality of our query suggestion model. In order to avoid a positional bias, we mix the provided suggestions during the evaluation phase. More in details, we have provided them the initial query along with 8 queries suggestions and 8 product suggestions. We have also provided some extra information, such as the category for which the query was originally executed to let them better understand the context and the meaning of the query. Judges could choose between three different values: relevant, irrelevant or no opinion (difficult to decide) for both types of suggestions.

For each query we have obtained two final rates, one for query suggestion and one for product suggestion. Those rates have been calculated using the majority vote among the three judges evaluations.

4.2 Evaluation measure

We evaluate the quality of the two types of suggestions provided by the model using Precision at rank N (P@N) and Mean Average Precision (MAP).
The precision at rank N, $P(N)$, for a specific query is defined as the percentage of relevant queries:

$$P(N) = \frac{\text{\# relevant queries}}{N} \tag{1}$$

$P@N$ is defined as the aggregated precision for all queries:

$$P@N = \frac{\sum P(N)}{\text{\# total queries}} \tag{2}$$

MAP is the average of Average Precision for all queries. For each query q_i:

$$AverageP(q_i) = \frac{\sum_j P(j) \cdot rel(j)}{\text{\# relevant queries}} \tag{3}$$

where the precision at rank j, $P(j)$ is defined as in equation 1, and $rel(j)$ is equal to 1 when the item at rank j is relevant, 0 otherwise. The mean average precision for a set of queries Q is the mean of the average precision scores for each query q_i:

Table 5: Overall scores for the query suggestions task expressed in terms of MAP and P@N, considering the top-3, top-5 and top-8 suggestions.

Dataset	MAP	P@1	P@3	P@5	P@8
Uncategorized	0.76	0.92	0.86	0.78	0.72
Categorized	0.81	0.94	0.89	0.81	0.75
Mobile phone	0.81	0.98	0.93	0.80	0.75
Oven	0.79	0.95	0.83	0.79	0.72
Fridge	0.80	0.98	0.91	0.83	0.77
Food supplements	0.59	0.82	0.64	0.59	0.30
Tires	0.34	0.70	0.29	0.21	0.09

Table 6: Overall scores for the product suggestions task expressed in terms of MAP and P@N, considering the top-3, top-5 and top-8 suggestions.

Dataset	MAP	P@1	P@3	P@5	P@8
Uncategorized	0.79	0.97	0.82	0.75	0.68
Categorized	0.82	0.95	0.87	0.80	0.71
Mobile phone	0.76	0.98	0.83	0.75	0.63
Oven	0.71	0.94	0.69	0.71	0.62
Fridge	0.80	0.98	0.79	0.71	0.64
Food supplements	0.59	0.81	0.64	0.49	0.34
Tires	0.39	0.76	0.41	0.25	0.07

$$MAP = \frac{\sum_{i=1}^{Q} AverageP(q_i)}{Q} \quad (4)$$

We also adopt a coverage metric to indicate for how many input queries the algorithm returns at least a minimum amount of query/product suggestions. We calculated coverage rate considering a minimum of 3, 5 and 8 queries.

4.3 Results

Evaluation metrics are calculated on three different type of datasets built as described in Section 4.1; First dataset, called "uncategorized", contains queries not related to any category; second dataset, called "categorized", contains queries and query results within a randomly chosen category; (iii) third dataset is formed by extracting queries from we 5 different categories: *mobile phones*, *ovens*, *fridges*, *food supplements* and *tires*. For this third dataset we refer to the associated set of queries using their category name.

The category datasets are extracted from categories that present substantially different characteristics and cardinalities. Table 1 shows, for each category, the total number of products belonging to the category, the number of products which have some matched offers, and the number of offers that are matched to products. Note that we did not chose only the most popular categories, but we also took into account categories having a low number of products (such as *food supplements*) in order to evaluate how the model deals with those less populated categories.

Coverage rate values reported in Tables 3 and 4 reflect the differences among the categories: the more populated categories reach higher results for both query and product suggestions. The exploitation of the category rank values in the "categorized" dataset enables the model to propose more results for both query and product suggestions, where the highest coverage values are achieved for the product suggestion task.

Tables 5 and 6 summarize the obtained results for the query suggestion model and the product suggestion model respectively. Results are shown in term of *MAP* and *P@N*, while varying *N* from 1 to 8.

From our experimental evaluation results we can conclude that our model proposes reliable suggestions on the top-8 result not only using the "categorized" dataset (75% query, 71% product), but also using the "uncategorized" dataset (72% query, 68% product). When referring to top-5 results precisions are around 80% for both datasets. Among categories, according to the coverage measure, the higher results are achieved for the more populated class.

Due to the similarity of the context of our work and the one from Zanon *et al.* (Zanon et al., 2012) and despite the fact that we used a different dataset, we propose a weak but but still significant compari-

son between our results and those achieved by their model. Unlike all the other works in literature, Zanon *et al.* (Zanon et al., 2012) evaluate their query suggestion model on two price comparison engines, reaching a coverage rate equals to 37% on the top-8 suggestion and an overall quality equals to 70%.

Our model overcomes those results, achieving significantly higher coverage rate: 75% on the "uncategorized" dataset and 56% on the "categorized" dataset. The quality of our query/product suggestion method is also higher than theirs for top-8 results. This confirms that a click-through based model better fits query suggestions task in price comparison and more in general in e-commerce website, as asserted in (Hasan et al., 2011).

5 CONCLUSIONS

In this paper we have presented a novel click-through-based query suggestion model specifically designed for price comparison websites.

Unlike most of the other generic query suggestion methods proposed in literature, to reach satisfying query/product suggestion results our approach takes advantage of most of the relevant information available for this category of website: product offers and product categories.

Similarly to other methods, these information are used to build a click-through bipartite graph. However, instead of using exploiting the association between queries and URLs clicked by users as in traditional click-through-based query suggestion approaches, our click-through bipartite graph stores associations between user queries and clicked products. Since in most price comparison websites product offers are clustered into products, our customized click-through bipartite graph allows our model to provide query and product suggestions based on products and not on direct URLs of product offers which may not be available anymore at the time that the suggestions is generated.

We evaluated our system both in terms of coverage rates and quality of the results for both types of generated suggestions (query and products), reaching high precision values, satisfying coverage rates, outperforming also the results of a competing recently published approach also specifically designed for the same query/product suggestion tasks for price comparison websites.

REFERENCES

Baeza-Yates, R., Hurtado, C., and Mendoza, M. (2004). Query recommendation using query logs in search engines. In *Proceedings of the 2004 International Conference on Current Trends in Database Technology*.

Boldi, P., Bonchi, F., Castillo, C., Donato, D., and Vigna, S. (2009). Query suggestions using query-flow graphs. In *Proceedings of the 2009 Workshop on Web Search Click Data*.

Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., and Li, H. (2008). Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Hasan, M. A., Parikh, N., Singh, G., and Sundaresan, N. (2011). Query suggestion for e-commerce sites. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*.

Jiang, D., Leung, K. W., Vosecky, J., and Ng, W. (2014). Personalized query suggestion with diversity awareness. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*.

Kato, M. P., Sakai, T., and Tanaka, K. (2013). When do people use query suggestion? a query suggestion log analysis. *Inf. Retr.*, 16(6):725–746.

Kim, Y. and Croft, W. B. (2014). Diversifying query suggestions based on query documents. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Lau, T. and Horvitz, E. (1999). Patterns of search: Analyzing and modeling web query refinement. In *Proceedings of the Seventh International Conference on User Modeling*.

Ma, H., Lyu, M. R., and King, I. (2010). Diversifying query suggestion results. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Mei, Q., Zhou, D., and Church, K. (2008). Query suggestion using hitting time. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*.

Meng, L. (2014). A survey on query suggestion. *International Journal of Hybrid Information Technology*, 7(6):43–56.

Ozmutlu, S., O. H. C. and Spink, A. (2003). Multitasking web searching and implications for design. In *Proceedings of the American Society for Information Science and Technology*.

ShoppyDoo (2015). `http://www.shoppydoo.it/`.

Song, Y. and He, L.-w. (2010). Optimal rare query suggestion with implicit user feedback. In *Proceedings of the 19th International Conference on World Wide Web*.

Song, Y., Zhou, D., and He, L.-w. (2012). Query suggestion by constructing term-transition graphs. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc.

Wen, J.-R., Nie, J.-Y., and Zhang, H.-J. (2001). Clustering user queries of a search engine. In *Proceedings of the 10th International Conference on World Wide Web*.

Wu, W., Li, H., and Xu, J. (2013). Learning query and document similarities from click-through bipartite graph with metadata. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*.

Zanon, R., Albertini, S., Carullo, M., and Gallo, I. (2012). A new query suggestion algorithm for taxonomy-based search engines. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*.

Zhang, Z. and Nasraoui, O. (2008). Mining search engine query logs for social filtering-based query recommendation. *Appl. Soft Comput.*, 8(4):1326–1334.