Product Retrieval for Grocery Stores

Petteri Nurmi[1], Eemil Lagerspetz[1], Wray Buntine[2], Patrik Floréen[1], Joonas Kukkonen[1]
[1]Helsinki Institute for Information Technology HIIT, Dept. of Computer Science, P.O. Box 68, FI-00014 University of Helsinki, Finland, firstname.lastname@cs.helsinki.fi
[2]National ICT Australia, 7 London Circuit, ACT 2601, Canberra, Australia, wray.buntine@nicta.com.au

ABSTRACT

We introduce a grocery retrieval system that maps shopping lists written in natural language into actual products in a grocery store. We have developed the system using nine months of shopping basket data from a large Finnish supermarket. To evaluate the system, we used 70 real shopping lists gathered from customers of the supermarket. Our system achieves over 80% precision for products at rank one, and the precision is around 70% for products at rank 5.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: [Retrieval models, Search process]; H.1.2 [**User/Machine Systems**]: [Human information processing]

General Terms

Algorithms, Languages, Experimentation

Keywords

Grocery Retrieval, User Evaluation, Retrieval Models

1. INTRODUCTION

More than 50% of customers create written shopping lists for major shopping visits [6]. Stores tend to use structured formats, e.g., product-category hierarchies (or taxonomies), that contain formal language, whereas customers tend to use a more free form natural language to describe items. As an example, a typical handwritten grocery list can contain everything from generic item descriptions (e.g., milk, juice) to very specific items (e.g., a specific package of washing powder).

The discrepancy between the way people create shopping lists and the way stores maintain product information raises the question of how to design an easy to use, intelligent mobile shopping assistant. While some design suggestions have been given in the literature [1, 2, 3], relatively little attention has been paid to the actual creation and use of shopping lists.

We describe and evaluate a grocery product retrieval system that can be used to map products expressed in natural language into products in a grocery store. Our system has been built using nine months of anonymized shopping data from a large Finnish supermarket (more than 12 million products bought). As the data is from a Finnish store, also the retrieval operates in Finnish. However, most of the techniques we employ are common information retrieval techniques, which suggests that our approach could also be used with other languages.

In addition to the shopping data, we have collected 132 real shopping lists from customers. Out of the lists, 62 were used to derive requirements for our retrieval system. The remaining 70 shopping lists were used to evaluate our system.

This work was supported in part by the Finnish Funding Agency for Technology and Innovation TEKES, under the project Personalised Ubiservices in Public Spaces, and by the IST Programme of the European Community, under the PASCAL Network of Excellence IST-2002-506778. The publication only reflects the authors' views.

2. RETRIEVAL SYSTEM

In the following we briefly describe the main functionality of our retrieval system.

Indexing

The retrieval system was constructed using nine months of shopping basket data. From the data, we constructed a database of product and category names, as well as indexes for words occurring in product and category names. We also stemmed the words and created indexes for stems appearing in product and category names. Before creating the indexes, units and package sizes were removed from product and category names. Stemming was performed using the Snowball Finnish language stemmer[1].

Index Expansion

In order to handle compound words, we use prefix and suffix index expansion. More specifically, we expand the index of every word j with product and category names that have a word that starts or ends with word j. For example, we add the product name *Vipset appelsiinitäysmehu* (Vipset orange juice) to the index of word *appelsiini* (orange) since *appelsiini* is a prefix of *appelsiinitäysmehu*.

Query Preprocessing

We lemmatize queries by removing partitive (singular and plural) and nominative plural case endings.

In order to handle colloquialisms, slang expressions and abbreviations, we constructed a lookup table. For example, *omppu* is a colloquialism that is mapped to *omena* (apple).

[1]`http://snowball.tartarus.org/algorithms/finnish/stemmer.html` [Retrieved 27-01-2008]

We expand queries with additional words when a query word is extremely common (e.g., *cheese*) or when the category name is a compound word that does not occur in any of the product names of the products in the corresponding category (e.g., *ananassäilyke (tinned pineapple, category)* vs. *ananasmurska, (mashed pineapple, product))*.

Words that span several product categories are considered stop words as they can cause the search to return a large number of potentially irrelevant products. Examples of stop words include manufacturer names and generic adjectives that describe characteristics of products.

Ranking

Items are ranked using an approximation of the posterior probability of an item given the query Q:

$$\begin{aligned} score(item, Q) &= \log p(item|Q) \\ &\propto \log p(item) + \log p(Q|item) \\ &\approx \log p(item) + \lambda \sum_j BM25(q_j), \end{aligned}$$

where $BM25$ is the value of the BM25 Okapi function for query term q_j and λ is a weight term. The BM25 function is given by [4]:

$$BM25 = \sum_j \log \frac{N - n_j + 0.5}{n_j + 0.5} \frac{(\alpha + 1) f_j}{f_j + \alpha\left((1-b) + bL\right)}.$$

The variables α and b are predefined constants. In the experiments we use Xapian[2] default values $\alpha = 1$ and $b = 0.5$. The variable f_j is the term frequency of word j, n_j is the number of product names in which term j appears, and N is the total number of product names. Finally, L is the normalized product name length, i.e., the length of the current product name divided by the average length of a product name. The variable λ was set to 0.75 on the basis of preliminary experiments.

Rank Augmentation

Intuitively, products that match both in product name and category name should receive higher ranks than other products. To achieve this behavior, we use a weighted extension of BM25 where the term and document frequencies are replaced with weighted linear sums [5]: $n'_j = dm_j + n_j$ and $f'_j = dv_j c_j + w_j f_j$.

Here d, v_j and w_j are weight terms. In the experiments, we use $d = 2$ and the weights v_j and w_j are set using edit distance calculations. Let p be an arbitrary word and let q_j be a query term. We set $w_j = 1.0 - d(q_j, p)/\max\{|p|, |q_j|\}$ where $d(\cdot,\cdot)$ is the edit distance between two strings and $|p|$ is the length of word p. The weights v_j are set similarly. We also use package sizes and stop words to augment rank scores. When the query contains a package size, we multiply the BM25 scores of matching products by d. If the product name matches a stop word that is part of the original query, we add the BM25 score of the stop word to the rank score.

Misspellings

To support misspellings, we maintain a distance index that contains word pairs and the edit distance between the two words. When the original query does not return any results, we consult the distance index using words within edit distance one. If this query does not return enough (ten or more) results, we use words within edit distance two.

[2]`http://xapian.org/docs/bm25.html`

N	Precisions	Evaluations
1	82.0%	843
2	77.9%	796
3	78.7%	785
4	72.9%	757
5	69.1%	751
MAP	71.4%	7454

Table 1: Results for precision at different ranks.

3. EVALUATION

Experiment Setting

In the experiment, we input the shopping lists into our search engine, one list at a time. The ten topmost results for each item in the shopping list were shown to an external evaluator who was asked to give positive or negative feedback on each result. The evaluator was also allowed to leave an item unrated if (s)he was not sure of the result. Each evaluator gave feedback on five shopping lists. The evaluators were recruited from students and staff of our department.

Results

The experiment resulted in a total of 7454 (subjective) relevance assessments. From the results, we calculated the mean average precision (MAP), as well as precision measures at different ranks (i.e., P@N). The resulting precision measures for the top 5 results are shown in Table 1.

We examined separately those queries whose top result was evaluated negatively. The main sources of negative feedback were (1) slang expressions that were not included in our lookup table, and (2) some complex queries where the product name was misspelled (e.g., query *ottermanni 17%*, result *Valio Oltermanni 1kg* – this was rated negatively because the package did not match). These two types accounted for 23 cases of negative feedback and when ignoring these assessments, precision at one equals 84.3%. Hence, the performance of our system could be increased by collecting more slang expressions, colloquialisms, abbreviations etc.

4. REFERENCES

[1] R. Bellamy et al. Designing an e-grocery application for a Palm computer: usability and interface issues. *IEEE Personal Communications*, 8(4):60 – 64, 2001.

[2] P. Kourouthanassis and G. Roussos. Developing consumer-friendly pervasive retail systems. *IEEE Pervasive Computing*, 2(2):32 – 39, 2003.

[3] E. Newcomb, T. Pashley, and J. Stasko. Mobile computing in the retail arena. In *Proc. SIGCHI*, pages 337 – 344, 2003.

[4] S. Robertson et al. Okapi at TREC-4. In *NIST Special Publication 500-236: The Fourth Text REtrieval Conference (TREC-4)*, pages 73 – 96, 1995.

[5] S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *Proc. 13th ACM CIKM*, pages 42 – 49, 2004.

[6] A. Thomas and R. Garland. Grocery shopping: list and non-list usage. *Marketing Intelligence & Planning*, 22:623 – 635, 2004.