Query Rewriting using Automatic Synonym Extraction for E-commerce Search

Aritra Mandal*
eBay Inc
San Jose, CA, United States
arimandal@ebay.com

Ishita Khan*
eBay Inc
San Jose, CA, United States
ishikhan@ebay.com

Prathyusha Senthil Kumar
eBay Inc
San Jose, CA, United States
prathykumar@ebay.com

ABSTRACT

Query rewriting is a critical component in modern search engines. It is the process of altering and enhancing raw user queries using synonymous keywords or structured metadata to improve search recall and relevance using data mining techniques applied on textual data and user behavioral signals. For example, the query *bicycle* is rewritten to match (bicycle OR bike) - i.e. all items that either contain the word *bicycle* or *bike* in their title are returned for this query. Choosing the right set of synonymous terms for a given query term is essential to ensure the quality of search results, especially in the context of e-commerce where buyer needs can be very specific. As an example, *shoe* is a good synonym for *shoes*, whereas *sandals*, while related, is not a good synonym for *shoes*. In this work, we describe one version of the approaches to query rewriting taken at eBay search. At a high level, we use a two step process to generate and apply synonyms for query expansions - 1. offline token level synonym generation and 2. runtime search query rewriting. In the offline phase, we first generate a large pool of candidate synonyms for query tokens using various techniques leveraging user behavioral data, inventory and taxonomy information and open source knowledge bases; then, we leverage a machine learned binary classifier trained on human judged binary relevance labels to filter the candidate synonyms that are truly useful as query expansions without compromising result set precision; this classifier allows us to leverage a wide variety of sources and techniques to generate synonym candidates by providing a scientific and scalable method to evaluate their effectiveness for query rewriting. This filtered set of token level synonyms is stored in a dictionary for runtime query rewriting. In the online phase, we rewrite user search queries by combining the token level synonyms in the dictionary, creating a boolean recall expression. We empirically demonstrate the value of this approach to enhance e-commerce search recall and relevance.

KEYWORDS

query rewriting, automatic query expansion, synonym generation, machine learning, big data, classification, web search, e-commerce

ACM Reference format:
Aritra Mandal*, Ishita Khan*, and Prathyusha Senthil Kumar. 2019. Query Rewriting using Automatic Synonym Extraction for E-commerce Search. In *Proceedings of ACM SIGIR Workshop on eCommerce, Paris, France, July 2019 (SIGIR 2019 eCom)*, 7 pages.

1 INTRODUCTION

One of the aims of query rewriting in information retrieval is to bridge the vocabulary gap between queries and documents. Other direct applications of rewriting a user query, such as addressing miss-spelling, different ways of describing the same entity regardless of level of formality exits. In the context of e-commerce search, user queries are much shorter and often colloquial, compared to product listing titles that are more verbose and contain formal terms. As an example, buyers may search for *noise absorbing blankets* when they are looking for *soundproof blankets*. In order to find all relevant inventory that matches the buyer's search intent, query terms are expanded or rewritten to synonymous terms which are also used to retrieve documents. In this example, we may rewrite the user's original query to *acoustic blankets*, *soundproof blankets* or *soundproof blanket* etc.

Several techniques have been proposed in literature to automatically identify query expansions including those based on a. pseudo-relevance feedback [22] where expansion terms are selected from the top results retrieved from a first round of retrieval, b. leveraging user search logs [3], [11], [10], [1], [17] which capitalize on the large volumes of readily available user behavioral logs to extract query substitutions using a variety of innovative techniques, c. ontology or thesaurus based [13], [16], [8] and [12], that employ either externally available knowledge bases or create one from the specific corpora to expand search queries. One more interesting approach proposed by Gao et.al [6][9][7] uses ranking mechanism to score rewrite lexicons specific to a query and boosts rewrites which are more relevant to the query.

The paper by Andrew et.al [20] gives an overall view of the search architecture at eBay. In our work we focus only on the query rewrite part of search at eBay. Our approach combines the effectiveness of several of the above mentioned techniques to extract synonymous words and phrases for query terms. Contrary to these techniques, we use a boolean recall basis to determine the items returned for a query, which are then ranked based on the user's specific ranking criteria (relevance, price, time on site, etc). The automatically generated query term synonyms are used to augment the query in the boolean recall expression. As an example, the query *mens bikes* is rewritten to *((men OR mens) AND (bike OR bikes OR bicycle OR bicycles)*, where the query term *mens* is expanded to *mens* and the term *bikes* is expanded to *bike, bicycle and bicycles.* These synonyms are mined using a number of techniques as described in 3.1 relying on the presence of these synonyms in behavioral logs and item index. These techniques result in a large pool of synonym

These two authors contributed equally.

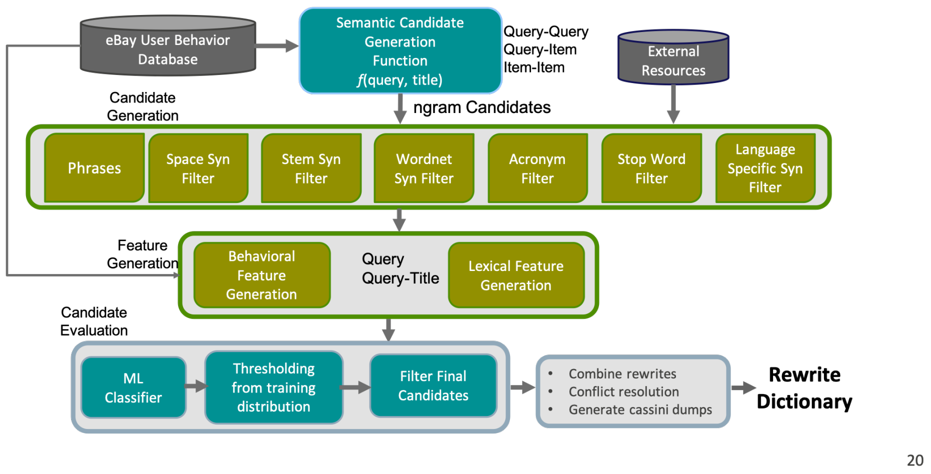


Figure 1: Schematic diagram showing the generation and evaluation framework of synonyms used in query rewrites

candidates with varying quality. A novel second step involving a machine learned classifier is used to differentiate the truly valuable synonyms from semantically related but not synonymous terms. As illustrated earlier, *shoe* and *shoes* are truly synonymous, whereas *shoe* and *sandals* are related but not synonymous. This classifier is trained using a variety of behavioral and lexical features on a human judged relevance label.

The rest of the paper is organized as follows: in Section 2 we provide an overview of our end to end process for e-commerce query rewriting, in Section 3 we describe the offline synonym dictionary generation process in detail, followed by evaluation and results for the synonym classifier and its effectiveness for query rewriting in Section 4 and conclude with future work in Section 5.

2 END-TO-END ARCHITECTURE FOR QUERY REWRITING

In this section we provide an overview of the synonym candidate generation and evaluation pipeline and the runtime use of keyword synonyms for query rewriting. Subsections 2.1 and 2.2 outline the offline and online components in our search query rewriting system, respectively.

2.1 Offline Synonym Candidate Generation and Selection Pipeline

The overall pipeline of the synonym candidate generation and evaluation framework for query rewriting is shown in Figure 1. We mine user behavioral data to collect search queries and item titles from search result pages (SRPs). From there, we use a semantic learning function that models three types of candidates:a. *query-query* transitions generated from pairs of queries within a single user session; b. *query-item* pairs of query and clicked item titles in the same SRP; c. *item-item* pairs of items shown in the same SRP. For each of these types of candidates, the semantic learning function essentially considers pairs with a certain minimum number of occurrences across search sessions, and then generates all possible token n-grams (n up to a fixed limit) from these pairs as potential synonym candidates. Next, we pass this exponentially large set of candidate pairs to a set of filtering components, one for each synonym generation technique (e.g. stemming, acronyms), that identifies if the candidate pair of n-grams qualifies as a synonym pair according to that technique. Following are some of the filtering components / synonym generation techniques that we will use in the scope of this paper: stemming equivalents, compounding equivalents, phrasing, open-source dictionary based synonyms and acronyms-full form equivalents (We are unable to disclose the full set of filtering components).

As mentioned earlier, we developed a machine learned classification model to evaluate the above generated synonym candidates, and filter out the synonym pairs that are truly valuable for query expansions. This classifier is optimized for a human judged binary relevance target (label=1 for candidates that are true synonyms, label=0 for candidates that are irrelevant or related but not synonymous). A number of behavioral and lexical features are extracted for the list of candidate synonyms being considered. In particular, for the behavioral features, we use clicks and other associated engagement signals from historic user sessions where the query token (trigger) and the corresponding synonym candidate occurred. Additionally we also derive ratio-based features from these count-based features. We construct lexical features based on semantics such as presence of numeric characters, lengths of the trigger and synonym etc. After the classifier has predicted a score for each candidate, we perform a score thresholding step to filter out the final set of synonyms. The thresholding is based on analyzing the score distribution on the training dataset in positive and negative classes and selecting a cutoff that balances between classifier performance and false negative rate. The final filtered query n-gram - synonym pairs are stored in a dictionary and used in runtime query rewriting to enhance the recall set for queries.

2.2 Runtime Query Rewriting using N-gram Based Synonyms

As mentioned above, once the final filtered synonym rewrites are generated for the n-gram query tokens, we store them in an offline rewrite dictionary for runtime query rewriting. At run-time, we have a two-phase approach to constructing boolean query rewrite expressions using the n-gram rewrite dictionaries: **query segmentation** and **query rewriting**. In the query segmentation phase, the raw user query is split into non-overlapping n-gram segments to maximize coverage in the rewrite dictionary. The segmentation heuristic used in this particular application is to take the longest matching segment from left among the query tokens, and then split the rest of the query tokens recursively in the same manner. In the query rewriting phase, we look up the synonyms for the n-gram segments in the synonym dictionary, and then combine them into a boolean expression. The different synonyms for a given segment are combined with an OR clause while the different segments for a given query are combined with an AND clause. For example, for the query *ps 4 games*, the first phase will segment the query as *ps 4* and *games* and the second phase might create a rewrite expression like *((ps 4 OR playstation 4) AND (games OR game))*, based on the following n-gram synonym rewrites: *ps 4 -> playstation 4* and *games -> game*.

3 SYNONYM DICTIONARY GENERATION FOR QUERY REWRITING

In this section, we deep dive into the steps involved in the offline synonym rewrite dictionary generation as outlined in Figure 1.

3.1 Mining Synonym Candidates

For generating individual synonym candidates, we process user behavioral data to collect query and item titles from SRPs. We use this data in a semantic learning function that models three types of transitions to generate potential synonym candidates. For each of these transitions, we create several pairs of n-grams (length upto a fixed n) as potential candidates as described below.

- **Query - Query Transitions:** generated from pairs of queries within the same user session. For example, consider a search session with the following queries: *womens dresses* and *ladies gown*. Using the pairs of queries within this set, we generate n-gram pairs as possible synonym candidates, where each member of the pair comes from a different query. With n=2 (bigrams), we generate the following candidates - *womens - ladies*, *womens - gown*, *dresses - ladies dresses - gown*, *womens dresses - ladies*, *womens dresses - gown*, *womens dresses - ladies gown*, *ladies gown - womens* and *ladies gown - dresses*.
- **Query - Item Transitions:** pairs of query and clicked item titles in the same SRP. Consider the query *womens dresses* - let's say we observe the following items being clicked in an SRP for this query - *womens dress/gown size 4*, *summer dress for women*. Similar to the query-query transition case, all pairs on n-grams between each pair of query and clicked-item are considered as potential candidates.
- **Item - Item Transitions:** pairs of items shown in the same SRP (same query). Using our example with query-item transitions, for pairs of items in the same SRP - *womens dress/gown size 4*, *summer dress for women*, we generate n-gram candidate pairs, one from each item.

For each of these types of candidates, we only consider pairs with a certain minimum number of occurrences across search sessions. Once generated, we pass this exponentially large set of candidate pairs to a set of filtering components, one for each synonym generation technique, that identifies if the candidate pair of n-grams qualifies as a synonym pair according to that technique. Following are some of the filtering components we currently have in our pipeline:

- **Stemming equivalents** Words that resolve to the same stem root, using Snowball stemmer [15]. E.g. *boat* and *boats*, both resolve to the same root *boat*.
- **Compounding equivalents** These are n-grams that when written as a compound word mean the same as the n-gram. These are more popular in some languages like German. E.g.: *sail boat* and *sailboat*, *granitpflastersteine* and *granit pflastersteine* or *granit pflaster steine*.
- **External dictionary based synonyms** We leverage open source knowledge bases and dictionaries to extract synonyms. One such example is WordNet [4, 5, 21] which is a large lexical database of English, where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms, each expressing a distinct concept. Each synonym in one group are interlinked by means of conceptual-semantic and lexical relations. From our n-gram candidates, we filter out synonyms as identified by wordnet. E.g.: *sunblock* and *sunscreen*, *tomahawk* to *hatchet*.
- **Acronyms-full form equivalents** We implement the Schwartz Hearst algorithm [18] for extracting acronym based synonyms that are identified using pattern matching on item titles and query item pairs. At a high level, the algorithm consists of two tasks: extraction of (short form, long form) pair candidates from text, and identification of the correct long form among the candidates in the sentence that surrounds the short form. E.g.: *hp* and *Hewlett Packard*, *NY&Co* and *New York & Company*.
- **Phrasing** By default, all tokens in a query are matched as a bag of words, meaning any item that contains those tokens in any order is returned for the query. For example, for the query *womens dress* we may return items like ***womens*** *summer* ***dress***. However, in some cases we might want to enforce strict word ordering while matching terms in items for user queries. This applies to named entities like brands, models, book names etc where the query intent is preserved only when matched exactly. Phrasing is a mechanism that enables us to do this enforcement. These are mined by understanding user preferences from query - item n-gram pairs. (Kumar et al. [19]). E.g.: If a user searches for *new balance shoes*, since new balance is a single entity, we enforce that only the items containing the words *new balance* contiguously in that order be returned for this query.

3.2 Synonym Selection with ML classifier

The different synonym mining techniques described in Section 3.1 yield a large pool of synonyms with varying degrees of quality and coverage. However, not all of these will be useful and could even potentially be harmful when use e-commerce search query expansion. For example, the words *tub* and *bathtub* may be used interchangeably in some contexts, but using one as an expansion for the other may result in some unexpected results from other product categories. We developed a Random Forest classifier [2] in order to filter out the truly useful synonyms from candidates that are related but not always synonymous.

3.2.1 ***Training Dataset and Classifier Information.*** Our random forest classifier for filter useful synonym candidates is trained on human judged binary labels indicating the applicability of the synonyms to query expansions (label=1 for candidates that are true synonyms, label=0 for candidates that are irrelevant or related but not synonymous). For example, the pair *shoe* - *shoes* will be labeled as 1, while the pair *shoes* - *sandals* will be labeled as 0. The training dataset itself is a weighted sample of synonym candidates from each synonym mining technique, where the weights are determined based on the quality of synonyms from each source - i.e. we oversample synonym types that have a larger fraction of negatives. For example, through our internal judgment of data quality, we empirically identified that synonyms based on external knowledge sources have a higher false positive rate than those from stemming, and hence we give a higher weight to synonyms from external sources compared to the ones from stemming. A training example pair of *trigger n-gram* and *synonym expansion* is labeled as relevant if and only if adding the synonym as an expansion to the original terms in the query n-gram brings in additional relevant inventory. The classifier has 100 trees with a tree depth of 20. We use the *sklearn* package from python[14] for the classifier operations.

3.2.2 ***Feature Engineering.*** A number of behavioral and lexical features (133 in total) are extracted for the list of candidate synonyms to be evaluated by the classifier. For behavioral features, we use clicks, sales and other associated engagement signals from historic user sessions (inclusive of SRP level data as well for query-item and item-item transitions) where the query terms (trigger) and the corresponding synonym candidate occurred. For this we sample queries and engaged items in a 2 week window. We are unable to reveal the full set of behavioral features used beyond the top 10. Below are examples of two of the behavioral features that appear in the top 10:

$$ExpansionInQuery^{Sale}_{TriggerInTitle} = \sum_{i=1}^{N}[ExpansionInQuery^{Sale}|TriggerInTitle]$$

For a given trigger and its expansion, the above equation calculates number of times a sale event happens when the trigger appears in an item title, and the expansion appears in the search query.

$$TriggerInQuery^{Click}_{BothTriggerExpansionInTitle} = \sum_{i=1}^{N}[TriggerInQuery_{Click}|BothTriggerExpansionInTitle]$$

Similarly, for a given trigger and its expansion, the above equation calculates number of times a click event happens when the trigger appears in the query, and both the trigger and the expansion appears in an item title. Here N is the number of query-title transitions considered. Additionally we also derive ratio-based features from these count-based features.

We also construct lexical features based on word semantics such as presence of numeric characters, string lengths, and lengths of matched and non-matched parts in trigger terms and expansions. To separate out the most predictive features we also added three random numbers features. For the final model we used all features with a feature importance greater than the first random feature, and ended up with 50 features in total.

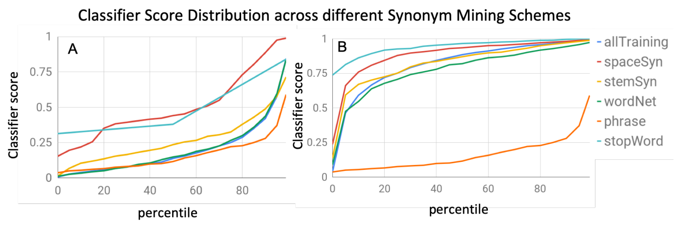


Figure 2: Classifier score distribution for A. negative class, and B. positive class

3.2.3 ***Threshold tuning for classifier scores.*** After the ML classifier has predicted a score for each synonym candidate mined through some of the processes detailed in Section 3.1, we perform a threshold tuning step to filter out the final set of good synonyms. The threshold selection is based on analyzing the score distribution on the training dataset in positive and negative classes and selecting a cutoff that balances between classifier performance and false negative rate. Figure 2 shows the analysis on the individual mining processes discussed in Section 3.1. With 1276 human labeled training samples in the English sites, we look at the distribution of random forest classifier scores for the negative(Figure 2A) and positive labels(Figure 2B) sub-sampled in space synonyms (27 positive labels, 240 negative labels), stem synonyms (70 positive labels, 153 negative labels), wordnet synonyms (200 positive labels, 131 negative labels), stop word synonyms (3 positive labels, 55 negative labels), and phrases (129 positive labels, 50 negative labels) and select a threshold of 0.3 where 85% of the negative labels and only 5% of the positive training labels are filtered out. The final filtered synonyms are stored in a dictionary that is used for runtime query rewriting.

4 EVALUATION RESULTS

In this section we discuss the experimental results for our approach. In Section 4.1, we share details about the synonym datasets, before and after filtering with the synonym classifier. In Section 4.2 we report metrics about the synonym classifier described in Section 3.2, and cover impact of synonym based query expansions on the number of search results returned across three of our major markets in Section 4.3 and results from our A/B tests in Section 4.4 and finally some anecdotal examples in Section 4.5.

4.1 Synonym Datasets

We filter the synonym candidates generated with various techniques described in Section 3.1 using a classifier trained on human judged labels, where a pair of query n-gram and its expansion is judged based on the utility of the expansion to bring in additional relevant items matching the query n-gram. For example, (**box**, **boxing**) $\longrightarrow$ **irrelevant**(**label** = **0**) because adding boxing as a synonym for box could change the intent from *box* as a *container* to *boxing* the sport, although they do qualify as stem synonyms, whereas (**rug**, **carpet**) $\longrightarrow$ **relevant**(**label** = **1**) because adding *carpet* as an alternate expansion to *rug* brings in additional relevant items for the query. We trained the synonym classifier using $4K$ training records from our three biggest markets (US, UK, and Germany) with 10 fold cross validation. Within this dataset, $1.8K$ are positive labels and $2.2K$ are negative labels. We also use 300 records as holdout set for final model testing. For generating final synonym candidates we sampled $2M$ synonym pairs, from which we filtered $940K$ synonym pairs using the ML classifier based synonym selection process as discussed in Section 3.2.

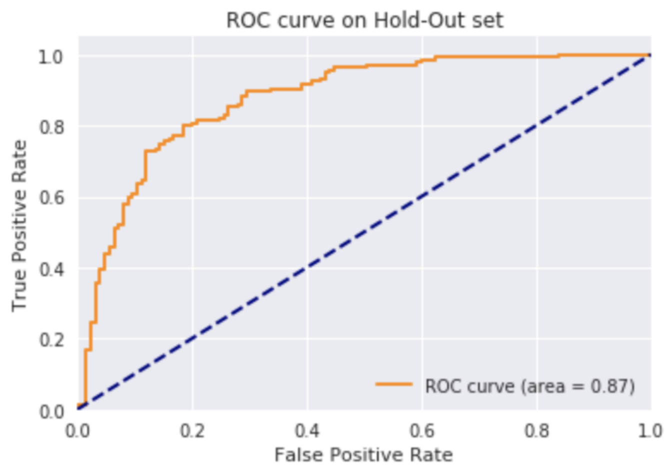


Figure 3: AUC and AUPRC of the model on the hold-out data

4.2 ML Classifier Results

In this section we discuss in detail the models performance on the holdout set. Figure 3 shows the AUC and AUPRC performance of the random forest synonym classification model on the holdout data. We achieved an AUC of 0.87 and an AUPRC of 0.84 on the holdout data.

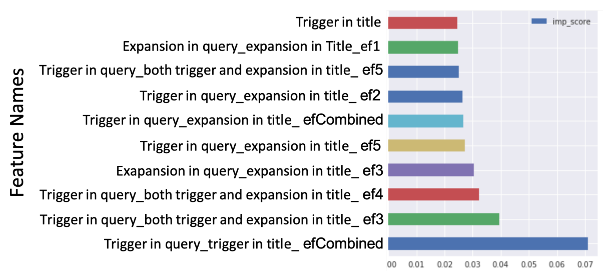


Figure 4: Top 10 features for the Random Forest synonym classifier; *ef* stands for Engagement Feature

Figure 4 shows the top 10 important features picked up by the model. We are unable to disclose the full list of features considered and the specifics of the different engagement signals. We indicate different engagement signals (denoted *ef*) in the Figure) with numbers when they are used as stand alone feature, and as *Combined* when all the different signals are combined to capture a certain feature. We observe that most of the top features picked up by the model are a combination of behavioral and semantic features. For example, the most important feature is an engagement signal capturing the fraction of that engagement signal for queries where the trigger n-gram is present in both the query and title, and the user behavioral signals are combined together. Another important signal is the number of items engaged with where the trigger n-gram was in the query and both the expansion and the trigger n-grams were present in the title of the engaged items.

Figure 5 shows the variation of prediction scores across different types of synonyms and the different transition types as discussed in section 3.1. For instance, we observe that for stemming based synonyms(stemsyn) the median classifier score is lower than the median score for compounding based synonyms(spacesyn). In Figure 6 we show the absolute error of the predictions made by the classifier across different rewrite types, and consistent with the previous observation, the absolute error for stemming based synonyms overall seems higher than that of the space synonyms for most subtypes.

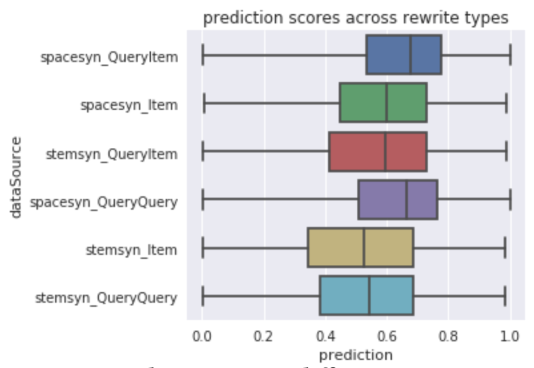


Figure 5: Prediction across different rewrite types

Another interesting observation we made is that the change in test error for the given model with increasing training data size saturates after 900 records. This suggests that with the given model complexity and feature set, a small training dataset has a reasonably high discriminating capability. Figure 7 shows the change in test error with increasing training data size.

4.3 Recall Impact of Query Rewriting

We sample queries across the three major sites and ran an experiment where we recorded the mean number of search results across the query set (recall size), with and without including these synonym based query expansions. The result is shown in Figure 8. This analysis shows the direct recall change we make in eBay search using synonym query expansions (on average 3x or more items are returned with synonym query expansions) for the impacted query set.

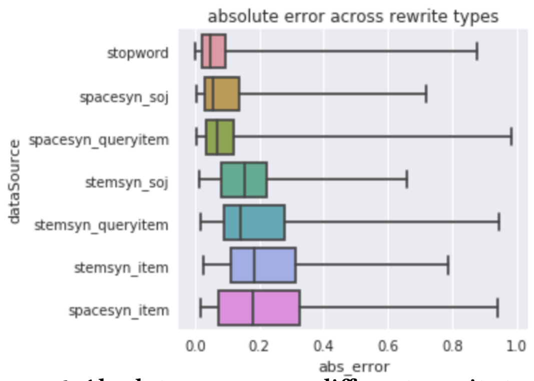


Figure 6: Absolute error across different rewrite types

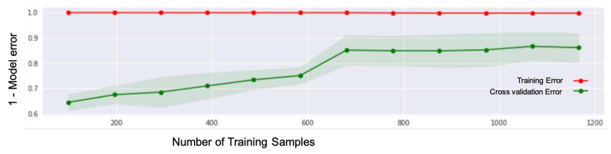


Figure 7: Absolute error across different rewrite types

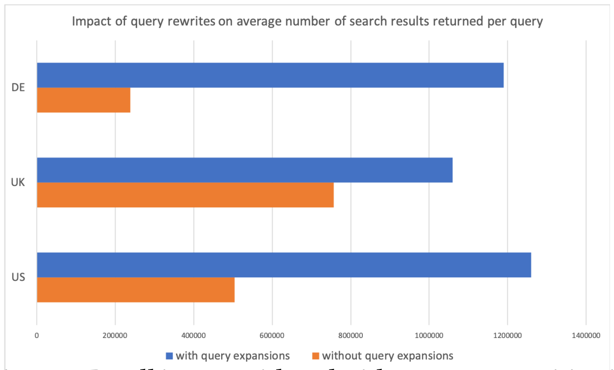


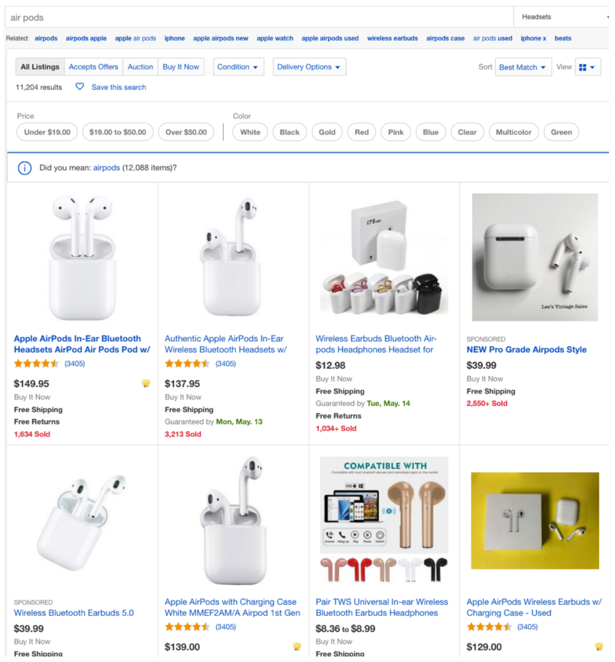
Figure 8: Recall impact with and without query rewriting in major sites

4.4 A/B Test Results

We ran online A/B tests to evaluate the value of the synonym mining and filtering techniques described in this work. For the test we used 33 percent of the live traffic for test and control, ran over a period of 3 weeks. We observed improvements in both relevance of the search result page (SRP) and user engagement across all markets, but are unable to share more specifics on the business metrics.

4.5 Anecdotal Examples

In this section we show an example of how good rewrites for a given term helps get more relevant results. The example shows the rewrite **airpods** for the term **air pods** in figure 9. As we can see 6 out of the 8 top listings for the query don't contain the original query term, and are present in the SRP due to the synonym expansion. All these results are relevant to the original query.



11,204 results

Did you mean: airpods (12,088 items)?

Apple AirPods In-Ear Bluetooth Headsets AirPod Air Pods Pod w/ (3405) $149.95 Buy It Now Free Shipping Free Returns 1,634 Sold

Authentic Apple AirPods In-Ear Wireless Bluetooth Headsets w/ (3405) $137.95 Buy It Now Free Shipping Guaranteed by Mon, May. 13 3,213 Sold

Wireless Earbuds Bluetooth Air-pods Headphones Headset for $12.98 Buy It Now Free Shipping Guaranteed by Tue, May. 14 Free Returns 1,034+ Sold

SPONSORED NEW Pro Grade Airpods Style $39.99 Buy It Now Free Shipping 2,550+ Sold

SPONSORED Wireless Bluetooth Earbuds 5.0 $39.99 Buy It Now Free Shipping

Apple AirPods with Charging Case White MMEF2AM/A Airpod 1st Gen (3405) $139.00

Pair TWS Universal In-ear Wireless Bluetooth Earbuds Headphones $8.36 to $8.99 Buy It Now Free Shipping

Apple AirPods Wireless Earbuds w/ Charging Case - Used (3405) $129.00

Figure 9: A example showing how adding air pods as rewrite to airpods improves recall

5 CONCLUSION

In this work, we presented our system for e-commerce query rewriting through token expansions. We described our end-to-end process for generating token synonym candidates, evaluating them and incorporating them for improving retrieval. We also shared our results in terms of the quality of the query rewrites filtered with a machine learned classifier when evaluated against a human labeled dataset as well as its value in improving the quality of search results. For future work, we plan to incorporate more techniques for mining synonym candidates, incorporating more context in the synonym selection process as well as leverage other sources like knowledge graphs for query rewriting.

REFERENCES

[1] Ricardo Baeza-Yates and Alessandro Tiberi. 2007. Extracting semantic relations from query logs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 76–85.
[2] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (Oct. 2001), 5–32. https://doi.org/10.1023/A:1010933404324
[3] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2002. Probabilistic query expansion using query logs. In *Proceedings of the 11th international conference on World Wide Web*. ACM, 325–332.
[4] Ingo Feinerer and Kurt Hornik. 2017. *wordnet: WordNet Interface*. https://CRAN.R-project.org/package=wordnet R package version 0.1-14.
[5] Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database.* Bradford Books.
[6] Jianfeng Gao, Xiaodong He, Shasha Xie, and Alnur Ali. 2012. Learning lexicon models from search logs for query expansion. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 666–676.
[7] Jianfeng Gao, Gu Xu, and Jinxi Xu. 2013. Query expansion using path-constrained random walks. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 563–572.

[8] Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan Cigarran. 1998. Indexing with WordNet synsets can improve text retrieval. *arXiv preprint cmp-lg/9808002* (1998).

[9] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to rewrite queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1443–1452.

[10] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. 2003. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology* 54, 7 (2003), 638–649.

[11] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 387–396.

[12] Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. 1999. Combining multiple evidence from different types of thesaurus for query expansion. In *SIGIR*, Vol. 99. 15–19.

[13] Roberto Navigli and Paola Velardi. 2003. An analysis of ontology-based query expansion strategies. In *Proceedings of the 14th European Conference on Machine Learning, Workshop on Adaptive Text Extraction and Mining, Cavtat-Dubrovnik, Croatia*. Citeseer, 42–49.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[15] Martin F. Porter. 2001. Snowball: A language for stemming algorithms. Published online. (October 2001). http://snowball.tartarus.org/texts/introduction.html Accessed 11.03.2008, 15.00h.

[16] Yonggang Qiu and Hans-Peter Frei. 1993. Concept based query expansion. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 160–169.

[17] Stefan Riezler and Yi Liu. 2010. Query rewriting using monolingual statistical machine translation. *Computational Linguistics* 36, 3 (2010), 569–582.

[18] Ariel S. Schwartz and Marti A. Hearst. 2003. A Simple Algorithm for Identifying Abbreviation Definitions in Biomedical Text. In *Pacific Symposium on Biocomputing*. 451–462.

[19] Prathyusha Senthil Kumar, Vamsi Salaka, Tracy Holloway King, and Brian Johnson. 2014. Mickey Mouse is not a Phrase: Improving Relevance in E-Commerce with Multiword Expressions. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*. Association for Computational Linguistics, Gothenburg, Sweden, 62–66. https://doi.org/10.3115/v1/W14-0810

[20] Andrew Trotman, Jon Degenhardt, and Surya Kallumadi. 2017. The architecture of ebay search. In *ACM SIGIR Forum. ACM*.

[21] Mike Wallace. 2007. *Jawbone Java WordNet API*. http://mfwallace.googlepages.com/jawbone

[22] Jinxi Xu and W. Bruce Croft. 1996. Query Expansion Using Local and Global Document Analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '96)*. ACM, New York, NY, USA, 4–11. https://doi.org/10.1145/243199.243202