Replicating Relevance-Ranked Synonym Discovery in a New Language and Domain

Andrew Yates[1] and Michael Unterkalmsteiner[2]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
ayates@mpi-inf.mpg.de
[2] Blekinge Institute of Technology, Software Engineering Research Laboratory, Karlskrona, Sweden
michael.unterkalmsteiner@bth.se

Abstract. Domain-specific synonyms occur in many specialized search tasks, such as when searching medical documents, legal documents, and software engineering artifacts. We replicate prior work on ranking domain-specific synonyms in the consumer health domain by applying the approach to a new language and domain: identifying Swedish language synonyms in the building construction domain. We chose this setting because identifying synonyms in this domain is helpful for downstream systems, where different users may query for documents (e.g., engineering requirements) using different terminology. We consider two new features inspired by the change in language and methodological advances since the prior work's publication. An evaluation using data from the building construction domain supports the finding from the prior work that synonym discovery is best approached as a learning to rank task in which a human editor views ranked synonym candidates in order to construct a domain-specific thesaurus. We additionally find that FastText embeddings alone provide a strong baseline, though they do not perform as well as the strongest learning to rank method. Finally, we analyze the performance of individual features and the differences in the domains.

Keywords: Synonym discovery · thesaurus construction · domain-specific search · replication · generalization

1 Introduction

The vocabulary mismatch problem [8] and its detrimental effect on recall [7] have long been recognized by the information retrieval community. In the absence of query expansion, whether by using pseudo relevance feedback or a lookup method, finding relevant information is a matter of constructing a query that expresses the user's information need using the same terms found in relevant documents. In the case of domain-specific search tasks, such as search in medical documents [12], legal documents [19], patents [27] and software engineering artifacts [15, 11], domain-specific synonyms complicate information retrieval as either the searcher or the retrieval engine need to be aware of terms that may be

used interchangeably. These specialized information retrieval tasks can benefit from thesauri that are crafted for their specific domain.

We chose to replicate and generalize a paper on using learning to rank for human-assisted synonym discovery, because we are interested in improving the synonyms in a classification system (ontology) from the building construction domain. While the users of the classification system are professionals, they are usually specialized in subsets of the construction business and need to coordinate with users specialized in other subsets of the domain. An improved set of synonyms would likely help the collaboration between the different parties using the classification system, such as when searching requirements documents.

Yates et al. [25] proposed a method that produces a ranked list of domain-specific synonyms using a domain-specific corpus as input. Their learning to rank approach uses a set of features that outperformed previous synonym discovery methods that relied on single statistical measures: pointwise mutual information over term co-occurrences [23] and pointwise total correlation between two terms and the syntactic context (dependency relations) in which the terms appear [10].

In this paper we replicate Yates et al.'s method on a different domain (building construction) and language (Swedish) in order to evaluate the generalizability of the approach. We used the original implementation, adapting preprocessing and features to the new setting. As such, we perform an inferential reproduction [9] where we draw similar conclusions as the original paper after evaluating it on a completely different dataset. Our contributions are *(1)* an evaluation of the method proposed by Yates et al. on a new language and different domain and *(2)* the proposal and evaluation of new features inspired by recent methodology advances and the differences introduced by the new domain and language.

2 Methodology

In this section we describe the synonym discovery task before describing our replication of the experimental setup used by Yates et al. We refer to the paper by Yates et al. as the original study and to this paper as the replication study.

2.1 Problem formulation

As in the original study, we define *synonym discovery* as the task of identifying a target term's correct synonyms from among a set of synonym candidates. Due to the difficulty of this problem, the original study argued that it is best approached as a synonym search task in which a domain-specific corpus is coupled with a learning to rank method in order to help a user quickly identify a target term's synonyms. For example, in the original study, the user might search for the target term *alopecia* with the intent of identifying domain-specific synonyms such as *hair loss* and *missing hair*. Such synonyms would ideally be ranked highly, but because of the task's inherent difficulty, incorrect terms like *greying hair* and *headache* are also likely to be ranked highly. This ranked list can then be used by a human editor in order to manually build a list of domain-specific synonyms.

For example, in the original study the method was used to augment a thesaurus mapping expert medical terms (e.g., *alopecia*) to lay synonyms often used in social media in place of the expert terms (e.g., *losing hair*).

The original study proposed to re-frame the evaluation of synonym discovery approaches from a TOEFL[3] style problem (i.e., given a term, pick the correct synonym from n candidate terms) to a ranking problem (i.e., given a term, evaluate how many true synonyms are ranked in the top X% of n candidates). This problem re-framing stems from the observation that the TOEFL style test represents the synonym discovery problem only when there is a sufficiently large number of synonym candidates. However, by increasing the number of incorrect choices, the evaluated approaches, including their own, were not able to answer the TOEFL-style question in most cases. Ranking synonym candidates according to their probability of being a true synonym of a target terms mirrors the synonym discovery problem more accurately. Ultimately, the involvement of a human editor is required to build an accurate domain-specific thesaurus.

One motivation for performing domain-specific synonym discovery is that we would like to cater for both propositional synonyms and near-synonyms. Propositional synonyms refer to terms that can be used interchangeably without affecting the truth condition of a statement. For example, the statements *He is a statesman* and *He is a politician*, referring to the same person, can both be true. This type of synonymity is concerned with identity rather than similarity of meaning, while near-synonyms refer to terms with similar meaning and are context-dependent [22]. For example, in the building construction domain, the term *kylelement (cooling panel)* and the term *förångare (evaporator)* are synonymous, and are used to describe a thermal cooling object. However, in the maritime domain, evaporators are used off-shore to produce fresh water, a different function from the building construction domain. These subtle domain-specific differences call for an approach that takes context into account and allows for the inclusion of human expertise when selecting near-synonyms from synonym candidates.

Formally, given a target term w_t and a set of candidate terms $\mathbb{C}$, a synonym discovery method ranks each candidate term $w_c \in \mathbb{C}$ with respect to its likelihood of being a synonym for the target term w_t in the target domain.

2.2 Replication design

The objectives of this replication study are to *(1)* evaluate the generalizability of the original finding that synonym discovery is best approached as a ranking problem, *(2)* to evaluate the generalizability of the learning to rank method proposed by Yates et al. in order to determine whether it is still the best approach in a new language and domain, and *(3)* to investigate whether the approach can be improved in our new setting by incorporating methodological improvements that were not considered in the original work (i.e., a language-specific feature, a contemporaneous term embedding feature, and a more sophisticated learning

[3] Test Of English as a Foreign Language

to rank model). We generalize over both the experimental setting (i.e., language and domain) and over time by considering both features specific to our new setting and new approaches that have become popular since the original work's publication. We evaluate the original approach, the best-performing baseline in the original work, a new embedding-based baseline, and a variant of the original approach using an improved learning to rank model on both the TOEFL task and the relevance ranking task from the original work.

2.3 Baselines

We compare the learning to rank method proposed in the original study with several baselines. We include the baseline that performed best in the original study as well two additional baselines based on embedding similarity and the similarity of dependency relation contexts.

PMI. The best-performing baseline in the original study was pointwise mutual information (PMI) calculated over term co-occurrences in sliding windows as proposed by Terra and Clarke [23]. We calculate PMI over 16-term sliding windows with the constraint that each window can only cover a single sentence. $PMI(w_c, w_t)$ can then be used as the ranking function for obtaining a ranked list of synonyms for the target term w_c.

EmbeddingSim. Word embeddings have become common since the original study's publication and are often used in information retrieval and natural language processing tasks. These methods are trained in an unsupervised manner on a large corpus to create dense word representations that encode some of the words' properties. The FastText [3], word2vec [17], and GloVe [21] methods are often used. Similar to the PMI method, word embeddings capture distributional similarity, and work has shown that much of their improvements over PMI come from the training setup used rather than from the underlying algorithm [13]. We train FastText on our corpus and rank a target term's candidates based on the cosine similarity between the term embeddings for w_t and w_c. The incorporation of this baseline is an example of "generalization over time" since this method is clearly relevant to the synonym discovery task, but it was not available at the time of the original study's publication.

LinSim. Lin's similarity measure [14] was originally proposed as a method for identifying synonyms and other related words. LinSim was used as a feature in the original study, but not as a separate baseline. This measure is similar to Hagiwara's methods [10] from the original study in that it considers pointwise mutual information over term contexts defined by dependency relations. It has the advantage of being less computationally expensive to compute on large corpora, however, so in this study we use it in place of Hagiwara's supervised and unsupervised methods.

2.4 Supervised Approaches

We evaluate two supervised learning to rank approaches on our dataset: a logistic regression as proposed in the original study, which is a pointwise method that

has been used for learning to rank in other contexts [26], and LambdaMART [6], a pairwise method. Given a target term w_t and a candidate term w_c, we compute the following features for use with both supervised methods:

- `Windows`: the number of windows containing both w_t and w_c, normalized by the smaller of the two counts. With the Wikipedia and Trafikverket corpora, a window is defined as a sequence of up to 16 terms appearing in a single sentence. With the Web corpus, a window is defined as a sequence of up to 16 terms appearing in a HTML element. Let $count_{win}(x)$ be the number of windows containing the term x and $count_{win}(x, y)$ be the number of windows containing both terms x and y. This feature is then calculated as

$$Windows(w_t, w_c) = \frac{count_{win}(w_t, w_c)}{min(count_{win}(w_t), count_{win}(w_c))}$$

- `LevDist`: the Levenshtein distance (i.e., edit distance) between w_t and w_c.
- `NGram`: the probability that the target term w_t appears in a specific position in a n-gram given that the candidate term w_c has also appeared in this position. As in the original work, we consider all trigrams that appear in our corpora. Let $count_{ng}(x)$ be the number of unique n-grams a term x appears in and $count_{ng}(x, y)$ be the number of unique n-grams in which both terms x and y appear in the same position (e.g., given the trigrams *rate/of/building* and *rate/of/construction*, both *construction* and *building* appear in the same position). This feature is then calculated as

$$NGram(w_t, w_c) = \frac{count_{ng}(w_t, w_c)}{count_{ng}(w_c)}$$

- `POSNGram`: the probability that the target term w_t appears in a specific position in a part of speech n-gram given that the candidate term w_c has also appeared in this position. As in the original work, this is equivalent to `NGram` after replacing each term with its part of speech.
- `LinSim`: the similarity between w_t and w_c as computed using Lin's similarity measure [14]. This measure requires dependency parsing, which we perform with MaltParser [18]. This feature also serves as one of our baselines.
- `RISim`: the cosine distance between the vectors for w_t and w_c, as computed using random indexing. We compute these vectors use the SemanticVectors package[4] [24] with its default parameters.
- `Decompound`: the number of components shared by w_t and w_c, normalized by the minimum number of components in either term. We use the SECOS decompounder [16] to split each term into their components, which decompounds each term using several strategies. We always choose the decompounding strategy that results in the largest number of components.
- `EmbeddingSim`: the cosine distance between word embeddings for w_t and w_c. We used FastText embeddings [3] trained on our corpus. This feature also serves as one of our baselines.

[4] https://github.com/semanticvectors/semanticvectors/

As described in the baselines section, we do not consider the two features from the original work that were based on Hagiwara's definition of contexts. The Decompound and EmbeddingSim features did not appear in the original work. We introduce the Decompound feature to account for the fact that many of our target terms are compound nouns; it is often the case that their synonyms share components with the target term. For example, the target term *apparatskåp* (*device cabinet*) shares the component *skåp* (*cabinet*) with its domain-specific synonym *elskåp* (*electrical cabinet*). We introduce the EmbeddingSim feature, on the other hand, because embedding-based similarity measures based on FastText [3], word2vec [17], and GloVe [21] have become popular alternatives to random indexing since the original work's publication.

3 Replication

3.1 Dataset

The original experiment focused on the medical side effect domain, with a corpus written in the English language. In this replication, we focus on the building construction business, in particular the provisioning and building of roads and public transportation infrastructure, and change the language to Swedish. We use the synonyms defined in CoClass [2] as the ground truth. CoClass is a hierarchical classification system, implementing ISO 12006-2:2015 [1], that is intended to facilitate the life-cycle management of construction projects. It is co-developed by the Swedish Transportation Agency (*Trafikverket*) and consultancy firms. Table 1 provides an overview of the dataset differences between the two studies.

Table 1: Comparison of the original experiment and the replication

	Original	Replication
Domain	Medical side effects	Building construction
Language	English	Swedish
Terms with/without synonyms	1,791/0	574/856
Average number of synonyms per term	2.8 ($\sigma = 1.4$)	3.8 ($\sigma = 4.4$)
Min/Max number of synonyms per term	2/11	1/46
Phrases/single term proportion	67%/33%	26%/74%
Corpus size (Number of documents)	400,000	4,241,509

While in the original study all terms were associated with at least 2 synonyms, only 574 of the terms in CoClass (40%) are associated with any synonym. Since our goal is eventually to improve the classification system with newly discovered synonyms, we did not remove synonyms that did not appear in the initially constructed corpus, which was crawled from Trafikverkets' publicly accessible document database (1,100 documents) and the Swedish Wikipedia (3,7 million articles). Only a subset of CoClass terms were found in this corpus, however.

Therefore, we devised the following strategy to construct a corpus: for each term in CoClass, we searched the public internet for this term using the Bing Search API[5], contributing 540,409 documents to the total corpus. Since each API call returns at most 50 hits, our budget was limited, and some terms in CoClass were common, we used a crawling strategy focused on identifying documents containing uncommon terms. More specifically, we restricted the number of crawled websites c based on the number of search results r for each term:

$$c = \begin{cases} 2500, & \text{if } r <= 10000 \\ 1000, & \text{if } 10000 < r < 100000 \\ 500, & \text{if } r >= 100000 \end{cases} \quad (1)$$

Category c_{500} contained 494 search terms, while c_{1000} contained 708 and c_{2500} contained 2261 search terms. Within c_{2500}, 528 search terms produced no hits at all. The search results demonstrate that the terminology in CoClass is very specialized as a large amount of terms were not even found on the publicly accessible internet.

3.2 Implementation Details

We preprocess our corpus using the efselab toolkit[6] [20] to tokenize and lemmatize the input text. In the case of the Wikipedia and Trafikverket corpora, we additionally perform sentence segmentation. On the Web corpus we use textract[7] to identify text inside of HTML elements (e.g., between and tags) and treat these text spans as sentences. We use efselab to perform part-of-speech tagging and MaltParser [18] to perform dependency parsing for the features that require this information. Our preprocessing differs slightly from the original work due to the changes in our input language and corpus. The original work used a tokenizer based on the Natural Language Toolkit[8] (NLTK), the Porter stemmer, NLTK's part-of-speech tagging, and RASP3 [5] for dependency parsing.

The original work took advantage of large, mature domain-specific thesauri to generate synonym candidates from the target domain. Such thesauri are not available in our language and domain, so we were forced to consider every term that appeared in our Wikipedia or Web corpora as a synonym candidate. We filtered these candidates by removing terms with a low term frequency, terms that did not appear much more often in our domain-specific Trafikverket corpus than in Wikipedia, and terms that were not tagged as a noun by our part-of-speech tagger. In particular, we required the candidates to have a TF of at least 300, to occur at least 30 times more often in Trafikverket than in Wikipedia, and to be tagged as a noun at least 50% of the time. These filtering steps

[5] https://azure.microsoft.com/en-us/services/cognitive-services/bing-web-search-api/
[6] https://github.com/robertostling/efselab
[7] https://textract.readthedocs.io
[8] https://www.nltk.org/

reduced the total number of synonym candidates from approximately 867,000 to 26,000 (97% reduction) at the cost of reducing the candidates' coverage of true synonyms in CoClass by approximately 26% (i.e., 74% of the synonyms remained after filtering). This left us with 290 target terms that both appeared in our corpus and had true synonyms in our candidate list.

As in the original work, we use the logistic regression implementation from scikit-learn[9] and scale the features to unit variance. We use the LambdaMART implementation from pyltr[10] with query subsampling set to 0.5 (i.e., 50% of the queries used to train each base learner) and the other parameters at their default values. For the word embedding feature, we used FastText[11] [3] to train 100-dimensional embeddings on our corpus using the skipgram method. FastText's other parameters were kept at their default settings.

Our code, the CoClass ground truth, and the URLs of documents in our corpus are available online.[12]

3.3 Experimental Setup

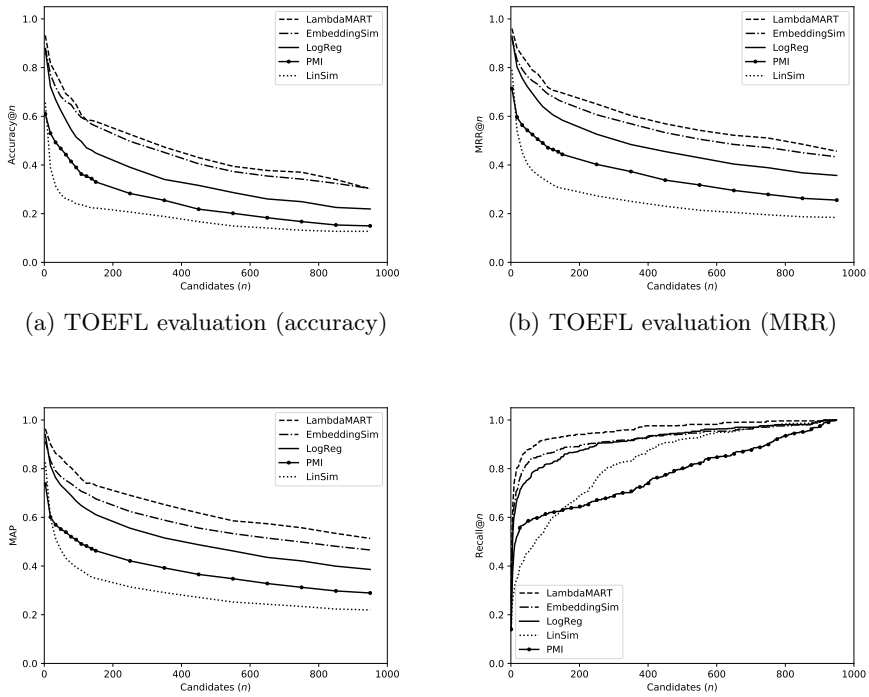
We conduct two experiments in which we compare the LogReg and LambdaMART learning to rank methods against three baselines: PMI, EmbeddingSim, and LinSim. In the TOEFL-style evaluation, we confirm the original study's conclusion that synonym discovery is best approached as a ranking problem. We then evaluate the methods' ability to produce useful ranked lists of synonym candidates in the relevance ranking evaluation.

Each method receives a target term and a set of candidates as input and outputs a ranked list of the candidates. In order to mirror the original study's evaluation, each target term is associated with up to 1,000 incorrect candidates that are randomly sampled from the full set of candidates $\mathbb{C}$. The supervised methods are trained with ten-fold cross validation. We create the cross validation folds based on target terms, so each target term appears in only one fold. We describe the metrics used by the two evaluations in their respective sections.

3.4 General TOEFL-Style Evaluation

As in the original work, we first perform a TOEFL-style evaluation to illustrate the difficulty of the domain-specific synonym discovery task. In this evaluation, methods are required to identify a target term's true synonym given one correct synonym candidate and n incorrect candidates. When $n = 3$ this corresponds to the TOEFL evaluation commonly used in prior work on discovering domain-independent synonyms. The original work made the argument that this evaluation is unrealistically easy and demonstrated that, in the consumer health domain, methods are unable to accurately identify synonyms when n is increased

[9] http://scikit-learn.org
[10] https://github.com/jma127/pyltr
[11] https://github.com/facebookresearch/fastText/
[12] https://github.com/andrewyates/ecir19-ranking-synonyms



(a) TOEFL evaluation (accuracy)

(b) TOEFL evaluation (MRR)

(c) Ranking evaluation (MAP)

(d) Ranking evaluation (Recall@n)

Fig. 1: Results on the TOEFL-style evaluation (top row) and relevance ranking evaluation (bottom row). While the top methods perform well in the TOEFL evaluation for low values of n, the performance of every method decreases as n is increased to more realistic values. The EmbeddingSim baseline performs well on its own, but is surpassed by the LambdaMART model that incorporates it as a feature.

to realistic values (e.g., $n = 1000$). In this section we repeat the general TOEFL-style evaluation in order to demonstrate that considering only $n = 3$ incorrect candidates is still unrealistically easy with our Swedish corpus focused on the building construction domain.

For each pair consisting of a target term and a correct synonym candidate, we randomly sample n incorrect candidates and feed the candidates as input to a synonym discovery method. As in the original work, we aggregate each method's predictions to calculate accuracy@n. We additionally report MRR (Mean Reciprocal Rank), which is a more informative metric because correct results in positions past rank 1 also contribute to the score. The result are shown in Figure 1a (accuracy) and Figure 1b (MRR). While LogReg, LambdaMart, and EmbeddingSim perform well at low values of n, their accuracy when approaching

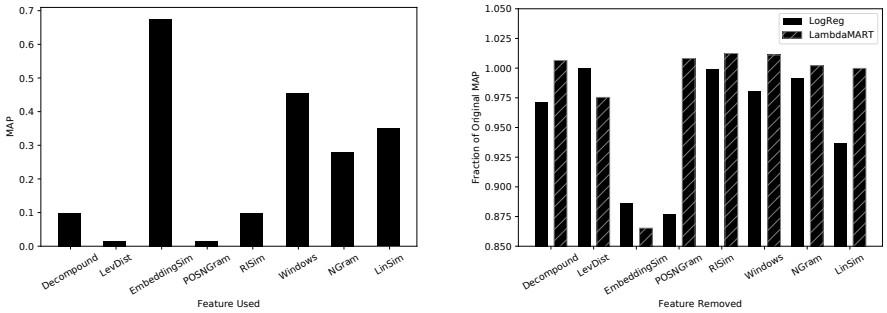
$n = 1000$ is less than 50%. The methods perform similarly in terms of MRR. While LogReg, the method from the original work, continues to outperform PMI and LinSim, the new EmbeddingSim baseline performs substantially better. This may be due to the fact that LogReg is a linear model and thus has difficulty weighting EmbeddingSim substantially higher than its other features. LambdaMART, the alternate learning to rank model evaluated in this work, is able to outperform both LogReg and EmbeddingSim. This illustrates that the synonym discovery task remains difficult in the new domain. For use cases where recall is important, such as ours, the task is best approached as a ranking problem.

3.5 Relevance Ranking Evaluation

In this section we evaluate the synonym discovery methods' ability to rank synonym candidates, so that they may be considered by a human editor. For each target term, we feed every synonym candidate as input to a synonym discovery method and calculate MAP (Mean Average Precision) and recall@n over the resulting rankings. In the context of this task, recall@n is the more interpretable metric: it indicates the fraction of correct synonyms that a human editor would find after reading through the top n results. The results are shown in Figure 1c (MAP) and Figure 1d (recall@n). In general the ranking of methods mirrors that from the TOEFL-style evaluation, with LambdaMART performing best. The top three methods perform similarly for different values of n, whereas PMI and LinSim perform differently. PMI performs better for low values of n, while LinSim begins to outperform PMI at roughly $n = 175$. As in the previous evaluation, the new LambdaMART and EmbeddingSim methods outperform LogReg. LambdaMART achieves 88% recall at $n = 50$, followed by EmbeddingSim with 82% recall and LogReg with 76% recall. This illustrates that the top performing methods can produce a useful ranking despite their low accuracy.

3.6 Feature Analysis

In this section we evaluate the contribution of individual features to the learning to rank models' performance. The MAP@150 achieved by each individual feature is shown in Figure 2a. EmbeddingSim performs substantially better than the other features. Windows, NGram, and LinSim are the only other features to achieve a MAP above 0.3, with Decompound, LevDist, and POSNGram performing poorly when used in isolation. In the original work, LevDist was the best performing single feature, with Windows, RISim, and the dependency context features (LinSim and Hagiwara) performing the next best. In our new domain, Windows and LinSim continue to perform well, but LevDist and RISim perform poorly. The difference in the domains and language may account for LevDist's decreased impact, since it is only a useful feature when synonym candidates have significant character overlap with target terms. EmbeddingSim was not included in the original work, but it is similar to RISim in that both methods are intended to capture distributional semantics.



(a) MAP when single features are used.

(b) MAP (fraction of max) when single features are excluded from the model.

Fig. 2: MAP@150 when only one feature is used (left) and the fraction of the entire model's MAP@150 when a single feature is excluded from the model (right). LevDist, the strongest feature in the original work, appears to have less utility in the new setting. Windows and LinSim, strong features in the original work, continue to perform well here.

We analyze the decrease in each model's performance when a single feature is removed in Figure 2b. The y-axis indicates each method's MAP as a fraction of the original MAP after a feature is removed. For example, removing the EmbeddingSim feature reduces the performance of both LambdaMART and LogReg to 85-90% of their MAPs when all features are used. With the exception of LevDist and EmbeddingSim, the LambdaMART model consistently achieves a smaller decrease in performance when any single feature is removed. As in the single feature analysis, EmbeddingSim is the best performing feature, and RISim does not contribute much to the models' performance. While POSNGram performed poorly in isolation, removing the feature decreases the performance of LogReg by approximately 12%, indicating that it is providing a useful signal used in conjunction with other features. Similarly, removing Decompound or LevDist decreases LogReg's or LambdaMART's performance by approximately 2.5%, respectively, despite the fact that they performed poorly as single features.

4 Generalizability

In order to better understand the generalizability of the learning to rank approach to synonym discovery, we discuss differences between this study and the original one in terms of the methodology and results.

Corpus Creation. Due to its focus on identifying synonyms of medical side effects, the original study used a corpus of 400,000 English forum posts related to health. In this study we focused on Swedish language synonyms in the building construction domain. This is a formal, specialized domain in comparison to health-related social media, which made it more difficult to identify documents

containing target terms or synonym candidates. To create a corpus with sufficient term co-occurrence information, we created a Swedish language corpus that was both larger and less homogeneous than the corpus used in the original study: 4.2 million Webpages from Wikipedia, the Swedish Transportation Agency (Trafikverket), and searches against the Bing API.

Preprocessing. While the preprocessing details differed in this study due to the change in languages, the techniques used were conceptually similar to those used in the original study. We use MaltParser in place of RASP3, and efselab's tokenization and lemmatizer in place of NLTK and the Porter stemmer.

Features and Method. We introduced two new features that were not present in the original study. Motivated by prior work that showed decompounding can improve recall in the German language [4], we introduced the Decompounder feature. This feature uses the SECOS decompounder [16] to split compound nouns into their components and measures the overlap between a target term's and candidate term's components. Figure 2b suggests that this feature slightly contributes to the performance of the LogReg method, but does not positively influence the performance of LambdaMART. EmbeddingSim, which computes the similarity between FastText embeddings, is the second new feature we introduced. FastText considers character n-grams when representing terms, which may make the Decompound feature redundant. We additionally introduced experiments on a new learning to rank model, LambdaMART, in order to compare its performance with the LogReg model used in the original work. We found that LambdaMART substantially outperformed LogReg, indicating the utility of using a more advanced model. We found that methods generally performed better on our domain and corpus. For example, LogReg achieved 50% recall@50 in the original study, whereas it achieves 76% recall@50 in this work. It is difficult to attribute these performance differences to specific factors, with the language, domain, and language register (i.e., professional language in this study and casual, lay language in the original study) all differing between the two studies.

5 Conclusions

In this work we replicated the synonym discovery method proposed by Yates et al. [25] in a new language (i.e., Swedish rather than English) and in a new domain (i.e., building construction rather than medical side effects). We found that in the new domain, the proposed LogReg method outperformed the PMI baseline as before. Motivated by methodological advances and the difference in languages, we introduced two new features and an alternate learning to rank method which we found to outperform the original approach.

These results provide evidence that *(1)* synonym discovery can be effectively approached as a learning to rank problem and *(2)* the features proposed in the original work are robust to changes in both domain and language. While our replication cannot provide evidence that the new EmbeddingSim feature works well in other settings, it does provide evidence that EmbeddingSim does not make the features used in the original work obsolete.

References

1. Building construction – Organization of information about construction works – Part 2: Framework for classification. Tech. Rep. 12006-2:2015, ISO (May 2015), https://www.iso.org/standard/61753.html
2. CoClass (Sep 2018), https://coclass.byggtjanst.se/en/about
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
4. Braschler, M., Ripplinger, B.: How effective is stemming and decompounding for german text retrieval? Information Retrieval **7**(3), 291–316 (Sep 2004)
5. Briscoe, T., Carroll, J., Watson, R.: The second release of the rasp system. In: Proceedings of the COLING/ACL on Interactive Presentation Sessions. COLING-ACL '06 (2006)
6. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Tech. Rep. MSR-TR-2010-82 (2010)
7. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Computing Surveys (CSUR) **44**(1), 1 (2012)
8. Furnas, G.W., Landauer, T.K., Gomez, L.M., Dumais, S.T.: The vocabulary problem in human-system communication. Communications of the ACM **30**(11), 964–971 (1987)
9. Goodman, S.N., Fanelli, D., Ioannidis, J.P.: What does research reproducibility mean? Science Translational Medicine **8**(341) (2016)
10. Hagiwara, M.: A supervised learning approach to automatic synonym identification based on distributional features. In: Proceedings Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Student Research Workshop. pp. 1–6. ACM (2008)
11. Haiduc, S., Bavota, G., Marcus, A., Oliveto, R., De Lucia, A., Menzies, T.: Automatic query reformulations for text retrieval in software engineering. In: Proceedings International Conference on Software Engineering (ICSE). pp. 842–851. IEEE (2013)
12. Kang, Y., Li, J., Yang, J., Wang, Q., Sun, Z.: Semantic analysis for enhanced medical retrieval. In: International Conference on Systems, Man, and Cybernetics (SMC). pp. 1121–1126. IEEE (Oct 2017)
13. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics **3** (2015)
14. Lin, D.: Automatic retrieval and clustering of similar words. In: Proceedings of the 17th international conference on Computational linguistics-Volume 2. pp. 768–774. Association for Computational Linguistics (1998)
15. Lucia, A.D., Fasano, F., Oliveto, R., Tortora, G.: Recovering traceability links in software artifact management systems using information retrieval methods. ACM Transactions on Software Engineering and Methodology (TOSEM) **16**(4), 13 (2007)
16. Martin Riedl, C.B.: Unsupervised compound splitting with distributional semantics rivals supervised methods. In: Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie. pp. 617–622. San Diego, CA, USA (2016)
17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

18. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E.: Maltparser: A language-independent system for data-driven dependency parsing. Natural Language Engineering **13**(2), 95–135 (2007)
19. Oard, D.W., Baron, J.R., Hedin, B., Lewis, D.D., Tomlinson, S.: Evaluation of information retrieval for e-discovery. Artificial Intelligence and Law **18**(4), 347–386 (Dec 2010)
20. Östling, R.: Part of speech tagging: Shallow or deep learning? Northern European Journal of Language Technology (NEJLT) **5**, 1–15 (2018)
21. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014), http://www.aclweb.org/anthology/D14-1162
22. Stanojević, M.: Cognitive synonymy: A general overview. Facta universitatis-series: Linguistics and Literature **7**(2), 193–200 (2009)
23. Terra, E., Clarke, C.L.: Frequency estimates for statistical word similarity measures. In: Proceedings Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology. pp. 165–172. AMC (2003)
24. Widdows, D., Cohen, T.: The semantic vectors package: New algorithms and public tools for distributional semantics. In: Semantic computing (icsc), 2010 ieee fourth international conference on. pp. 9–15. IEEE (2010)
25. Yates, A., Goharian, N., Frieder, O.: Relevance-ranked domain-specific synonym discovery. In: Advances in Information Retrieval - 36th European Conference on IR Research. ECIR '14 (2014)
26. Zeng, H.J., He, Q.C., Chen, Z., Ma, W.Y., Ma, J.: Learning to cluster web search results. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 210–217. SIGIR '04 (2004)
27. Zhang, L., Li, L., Li, T.: Patent mining: a survey. ACM SIGKDD Explorations Newsletter **16**(2), 1–19 (2015)