Automatic Boolean Query Refinement for Systematic Review Literature Search

Harrisen Scells
The University of Queensland
Brisbane, Australia
h.scells@uq.edu.au

Guido Zuccon
The University of Queensland
Brisbane, Australia
g.zuccon@uq.edu.au

Bevan Koopman
CSIRO
Brisbane, Australia
bevan.koopman@csiro.au

ABSTRACT

In the medical domain, systematic reviews are a highly trustworthy evidence source used to inform clinical diagnosis and treatment, and governmental policy making. Systematic reviews must be complete in that *all relevant* literature for the research question of the review must be synthesised in order to produce a recommendation. To identify the literature to screen for inclusion in systematic reviews, information specialists construct complex Boolean queries that capture the information needs defined by the research questions of the systemic review. However, in the quest for total recall, these Boolean queries return many non relevant results.

In this paper, we propose automatic methods for Boolean query refinement in the context of systematic review literature retrieval with the aim of alleviating this high recall-low precision problem. To do this, we build upon current literature and define additional semantic transformations for Boolean queries in the form of query expansion and reduction. Empirical evaluation is done on a set of real systematic review queries to show how our method performs in a realistic setting. We found that query refinement strategies produced queries that were more effective than the original in terms of six information retrieval evaluation measures. In particular, queries were refined to increase precision, while maintaining, or even increasing, recall — this, in turn, translates into both time and cost savings when creating laborious and expensive systematic reviews.

KEYWORDS

Systematic Reviews, Query Formulation, Boolean Queries, Query Transformations

ACM Reference Format:
Harrisen Scells, Guido Zuccon, and Bevan Koopman. 2019. Automatic Boolean Query Refinement for Systematic Review Literature Search. In *TheWebConf '19: The Web Conference.* ACM, New York, NY, USA, 11 pages.

1 INTRODUCTION

Systematic reviews are highly reliable sources of evidence, created by synthesising all relevant studies in a comprehensive literature review for a highly focused research question. Medical systematic reviews are held to a particularly high standard because of the role they play in both evidence-based medicine and health policy as a whole. Systematic reviews both inform how medical professionals diagnose and treat patients, and how governing bodies decide upon health management and policies [14].

With the rise of web-based databases of medical literature, now containing tens of millions of medical citations (often in the form of abstract, title, MeSH headings and metadata, which reference full studies of, for example, randomised controlled trials), it is becoming increasingly difficult and highly costly to identify relevant literature to include in a systematic review [29]. For example, one of the most popular web-based databases, PubMed, contains approximately 28 million citations of medical studies. While recall is paramount in that all potentially relevant citations must be retrieved, precision in the case of systematic review literature retrieval is still critically important. Systematic reviews can cost upwards of a quarter of a million dollars [17] and can take several months to complete. A large portion of this cost can be attributed to the screening phase, where there are typically many false positives (i.e., citations retrieved which are not relevant to the research question of the systematic review). For systematic reviews of highly specific areas of medicine, finding relevant studies requires a highly specific query which cannot be too broad (which minimises the false positive rate). For broad studies such as scoping reviews, the query cannot be too specific as to limit the number of relevant citations that are not retrieved (i.e., minimising the false negative rate). This balance of potentially relevant and potentially non-relevant citations is currently managed by the intuition and knowledge of highly trained and skilled information specialists who are deeply familiar with the search systems in which they formulate their queries, and often, but not always, also with the topic of the systematic reviews they are formulating queries for.

The queries that information specialists formulate are highly complex Boolean queries. Systematic review literature search is dependent on, and has always used, Boolean retrieval for two reasons: (i) the Boolean query syntax enables specialists searchers to have fine-grained control over exactly which citations are retrieved, and (ii) the Boolean query which is published alongside the review, is used for reproducibility purposes; for example, when the recommendation of a systematic review is in doubt or a review needs to be updated. Ultimately the Boolean query controls the outcomes and decisions of a systematic review.

This paper investigates automatic methods for refining queries used to retrieve literature for systematic reviews. Prior work by Scells and Zuccon has investigated syntactic transformations of the Boolean query, and have empirically shown that these could improve the effectiveness of the queries originally formulated by expert information specialists [24]. In their approach, a query was modified at the Boolean query language level; for example, by

TheWebConf '19, 2019, San Francisco, USA

changing an *OR* operator into an *AND* operator for a specific Boolean clause. These transformations did not add or remove the original query terms; i.e., they did not perform query expansion or reduction, apart from making use of, or removing, the Medical Subject Headings (MeSH) explosion operator (which allows for the retrieval of citations that contain MeSH concepts subsumed by the concept specified in the query). In this work, we investigate whether semantic transformations (query expansion and reduction by drawing upon semantically similar terms) can further improve the effectiveness of the Boolean queries. Specifically, the context of this work is to refine existing Boolean queries that have been created for the explicit purpose of systematic review literature search. We define a good query refinement process as one where fewer citations are retrieved, but the number of relevant citations stays the same (or with very minor, tolerable reductions). In Information Retrieval terms, this equates to increasing precision, while maintaining or even improving recall. In the case where recall is improved, note that in the case of systematic reviews, a number of studies that are used to synthesise results are often found *outside* the search process, either by snowballing references (i.e., finding relevant studies by examining the references of retrieved citations), or having prior knowledge of the existence of relevant studies.

The methods proposed in this paper are envisioned not to replace the experience and knowledge of human searchers but, instead, to assist them formulate more effective queries (e.g., explicitly recommending more effective queries). To investigate this topic, we pose the following research questions:

RQ1: Which types of transformations are the most effective at refining Boolean queries in systematic review literature search?

RQ2: What impact do unjudged documents have on the effectiveness of refined Boolean queries in systematic review literature search?

To answer **RQ1**, we first devised a set of strategies to generate semantic and syntactic transformations within the Query Transformation Chain Framework of Scells et al. [24]. These were then applied to the original Boolean queries in an iterative fashion so as to generate a set of transformed query candidates. The generated candidate queries were then represented within a feature space, which was then used to automatically select a transformed Boolean query that was predicted to improve over the original query (if no improvement was predicted, then original input query was retained). We then selected the top query from the set of candidates with a query candidate selector function which maximised a target evaluation measure. The effectiveness of the query refinement was determined by comparing the original queries with the ones that were automatically refined.

To answer **RQ2**, we perform evaluation by considering the unjudged documents that may be retrieved by the automatically refined queries. We do so by considering two methods for computing which portion of the unjudged citations (i.e. the residual) should be used for computing relevance. The methods outlined for **RQ1** and **RQ2** are explained in detail in Section 3.

2 RELATED WORK

The modification or transformation of a query, such as query expansion and query reduction, but not limited to any semantic and syntactic modification (e.g., via synonyms of a term, adding related terms, removing unnecessary terms, correcting spelling mistakes) has been shown to significantly improve the effectiveness of a query [3, 6, 18, 30]. Previous work, however, has focused almost exclusively on non-Boolean queries (i.e., keyword queries).

In the professional search setting, Kim et al. [10] was able to generate Boolean queries using a decision tree method and then automatically suggest queries. A decision tree is generated by selecting unigrams from the top-k pseudo-relevant documents and nodes in the tree are selected with a probability of that term appearing or not in the pseudo-relevant set (leading to conjunction and negation). Queries are then suggested by training a learning to rank model on query performance predictor (QPP)-based features. Both general-purpose QPPs and Boolean query-specific QPPs are used. When general-purpose QPPs are used, only the query terms that are associated with conjunctions are considered. In the context of systematic reviews, Scells et al. [22] have shown that QPPs were not effective in predicting the performance of queries in this domain.

Karimi et al. [9] have found that simplifying field restrictions and operators, and removing clauses from keyword queries (not Boolean) provides significant improvements in recall for systematic review queries, while precision was slightly degraded. Keyword queries, however, have a major limitation: they do not capture complex information needs by restricting sets of documents; i.e., they operate in a best-match setting where ranking is important[1]. Another perspective is that keyword queries do not allow the information specialist explicit control over the set of documents retrieved.

Because Boolean queries are so deeply engrained in the scientific methodology of constructing systematic reviews, we focus on modifications or transformations that can be applied to Boolean queries. Scells et al. have also investigated the modifications to queries in a systematic review setting in two studies. One study found that integrating clinical labels into the retrieval model (i.e., annotating both queries and documents) significantly improved precision while slightly degrading recall [25]. The other study focused on improving existing Boolean queries with automatic candidate generation and selection [24]. While Scells et al. [24] use only syntactic transformations, we advance the state-of-the-art by integrating syntactic *and* semantic transformations of queries and then automatically selecting the best candidate.

3 METHODOLOGY

3.1 Query Transformation Chain Framework

A query transformation chain is the repeated application of one or more transformations to a query which results in a set of variations of the original query. A simplified example of a query transformation chain is shown in Figure 1, where three transformations (τ_1, τ_2, and τ_3) are applied to three clauses (c_i) of the original query q. This process of applying transformations to clauses of a new query continues until the stopping criteria of the candidate selection function is met. After the transformations are applied, a final, rewritten query $\hat{q}$ is produced. A query chain can either be used to generate all possible variations of a query given the set of transformations,

[1] Although another task in this domain – screening prioritisation – would benefit from this.

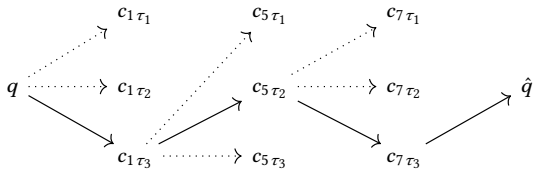

Figure 1: A query q transformed via a query transformation chain into a new query $\hat{q}$. Three transformations (τ_1, τ_2, and τ_3) are applied to clauses (c_i) of a query at each point in the chain. The path of each transformation chosen through the chain is indicated by a solid line. Only the path of transformations taken for $\hat{q}$ are shown.

or at each application of transformations, a candidate is selected. This candidate selection process is described in detail in Section 3.4.

3.2 Query Clause Transformations

Scells et al. [24] considered only syntactic transformations of a query. We extend that work by integrating several semantic transformations. Syntactic transformations are those that modify aspects of the query unrelated to the structure of a query: logical operators or the fields keywords are restricted to. The other form of transformation is semantic, whereby aspects of a query are modified structurally (through the addition or removal of clauses). In this work, we use six types of transformations in total. Of these, two are syntactic and four are semantic. In addition, four of the transformations can be applied to non-atomic clauses and three can be applied to atomic clauses. The six transformations in total we use in this work are presented in Table 1 with the classification (syntactic or semantic) and the applicability of the transformation (to atomic or non-atomic clauses). Each transformation is described in detail below:

(1) **Logical Operator Replacement** The logical operator replacement transformation syntactically replaces the Boolean operators in a query. For example, `(A **AND** B)`→`(A **OR** B)`. This transformation only considers `AND` and `OR` operators.
(2) **Field Restrictions** The field restrictions transformation syntactically modifies the fields an atomic clause is restricted to in Boolean search. For example,

```
(cancer[Title] AND lungs[Title])
                ↓
(cancer[Title/Abstract] AND lungs[Title])
```

This transformation produces the following modifications to fields: `title` to `abstract`, `title` to `title & abstract`, `abstract` to `title`, `abstract` to `title & abstract`, `title & abstract` to `title`, and `title & abstract` to `abstract`.
(3) **MeSH Explosion** MeSH is a medical ontology organised in a tree structure. Explosion is the subsumption of all nodes underneath a particular term node in the ontology. MeSH explosion can be considered a semantic transformation in that it implicitly adds terms to a query. This transformation toggles if a MeSH term should be exploded or not. Systematic review query languages all support some way for explosion to be toggled, normally in the retrieval model, thus not explicitly modifying the semantics of a query.

Transformation	Classification	Applicability
Logical Operator Replacement	Syntactic	Non-Atomic
Field Restrictions	Syntactic	Atomic
MeSH Explosion	Semantic	Atomic
MeSH Parents (new)	Semantic	Atomic
Clause Removal (new)	Semantic	Atomic
cui2vec Expansion (new)	Semantic	Atomic

Table 1: Classification (semantic or syntactic) and Applicability (transformation can be applied to atomic or non-atomic clauses) of transformations used in this work.

(4) **MeSH Parents** The MeSH parents transformation acts in the opposite direction of the explosion transformation: moving up one level in the tree (if possible), thus selecting a broader MeSH term. For example, if the MeSH term is `Skull Neoplasms`, then this transformation selects the parent term `Bone Neoplasms`.
(5) **Clause Removal** The clause removal transformation semantically modifies a query by removing a non-atomic clause from a query. This transformation will remove atomic clauses from a non-atomic clause until there are no more atomic clauses left (at which point the non-atomic clause that groups them is removed).
(6) **cui2vec Expansion** The cui2vec transformation is a semantic query expansion transformation that uses a pre-trained set of clinical concept embeddings [1] to find related keywords. [2] We first map keywords to concepts using QuickUMLS [27], restricting matches to preferred candidates. Next we find the top-k similar concepts (which are then used as expansions E) using the clinical concept embeddings via cosine similarity; scores are then normalised using the softmax function. A mapping derived by Jimmy et al. [7] is used to map concepts to their most common string. In our experiments, we expand to a maximum of five candidates. Finally, to integrate the new expansions into the Boolean query, we replace the original keyword with a logical `OR` operator that groups the expansions. The fields of the expansions are inherited from the original keyword. Finally, the original keyword that was used for expansion is then added to the atomic clauses in the new non-atomic clause.

3.3 Query Candidate Features

The ranking of query candidates requires discriminative features in order to select the most effective query. We use 39 features in total, described in Table 2. Two notable features which we consider to be novel are Δ, which computes the difference between measurements indicated in Table 2 with †, and ω_i which represents the ith transformation applied to a query in the query chain process. For performance and effectiveness reasons we do not include general purpose QPPs (Query Performance Predictors) as in similar

[2] As the name suggests, the embeddings are for clinical *concepts* (rather than terms), identified as **Concept Unique Identifiers** (CUI) from the UMLS Metathesaurus. A single CUI may represent different names or *strings* in different source vocabularies of the UMLS Metathesaurus.

work [10]. This is because: (i) most QPP measurements rely on costly retrieval statistics which becomes prohibitive with the number of queries that are generated, and (ii) previous work has shown that QPPs are often not correlated with retrieval effectiveness [5, 22] and thus may hamper the quality of ranking.

3.4 Query Candidate Selection

Query candidate selection is the process by which the next candidate to transition to in the query chain is chosen. The formalisation of candidate selection is described as follows. Following the generation of candidate queries $\hat{\mathbb{Q}}_q$ from the initial query q, the next candidate q^* is selected as the next transition in the query chain via the maximisation of function $f(\hat{q})$ where:

$$q^* = \underset{\hat{q} \in \hat{\mathbb{Q}}_q}{\text{argmax}}\, f(\hat{q}) \tag{1}$$

The candidate generation and selection process is repeated until the stopping criteria is met. Here, we define the stopping criteria with two objectives: (i) the query chain reaches a depth of five as in previous research [24], and (ii) the number of retrieved citations for a candidate query is zero.

When generating queries for training purposes, we only considered the stopping criteria when the depth of a chain reached the maximum length of five. We did this so that we would have sufficient negative training examples. In practice, this resulted in a severely imbalanced training set with the majority of examples being negative and only a small portion positive. In training models to maximise the candidate selection objective function, we discovered empirically that removing the majority of negative examples (i.e.,from rebalanced the positive and negative examples via negative sub-sampling) increased the effectiveness of ranking.

The maximisation function $f(\hat{q})$ can be learnt as a machine learning problem. In this work, we learn $f(\hat{q})$ as both a learning to rank task (similarly as in [24]) and a nearest neighbour task (new to this work). For both tasks, we train a different model for each evaluation measure that we seek to optimise. For the learning to rank task, we use DART: an ensemble of boosted regression trees which incorporates dropouts [21]. In our empirical testing we found this method to be the best for this task when compared to other state of the art techniques such as LambdaMART. For the nearest neighbour task, we create a model which records the most effective query in each topic (given an evaluation measure), and the distance to and reduction in effectiveness for all other queries in that topic. For an unseen set of candidate queries, each target query in the set is ranked by minimising the distance and maximising the score of the queries in the aforementioned model.

3.5 Sampling Query Candidates

In generating training data for automatic query candidate selection methods, we encountered an exponential growth in queries generated. This growth is computationally undesirable: for example, if on average 100 candidates are generated for a query, and the total length of a chain is 5 then a total of 10^9 queries would be generated (n^d where n is the number of candidates and d is the depth). Additionally, if on average it takes 30 seconds to generate a set of queries and features, then for that query it would take over 950 years to complete. For this reason, we sample queries to cut the total number of queries that are included for training. Note that the $O(n^d)$ complexity does not affect testing as we are not exploring the space of transformations for a query, rather we are ranking candidates, selecting the top-1 and generating transformations for that. Our sampling is as follows. For a seed query q, we generate a set of candidate queries $\hat{\mathbb{Q}}_q$. From this set, we take a minimum of 20 candidate queries and if there are more than 20 queries generated, we take a further 10%. We then cut further by removing any identical queries that have already been processed. Finally, of the queries that are generated from the previous step, we take 10% of these new queries and repeat the process.

3.6 Evaluation

We perform evaluation as a standard ad-hoc search task. Relevance assessments are obtained from the studies that are reported in each systematic review as *included* (relevant), and *excluded* (non-relevant). Queries obtained through query refinement may retrieve citations that were not assessed by the reviews (i.e., unjudged); thus, relevance assessments are incomplete. At the first stage we assume all citations which were not judged as non-relevant. Remember, however, that unlike in traditional ad-hoc information retrieval evaluation, no pooling using a large variety of systems was executed: only one query and one system contributed to the relevance assessments — other, non retrieved but relevant citations may very well exist. Thus, later in the analysis of the results, we use two heuristics to bound the effect of unjudged documents on the effectiveness of the studied methods. To this aim, we use and further extend the intuition of residual analysis described for the rank bias precision measure [19]:

3.6.1 Optimistic Residual. The first heuristic consists of assuming all unjudged citations are relevant, as done for the computation of the heuristic in rank bias precision [19]. This value provides the upper bound on the value of the evaluation measure that would be obtained if all citations were assessed.

3.6.2 Maximum Likelihood Residual. The second heuristics attempts to provide a stricter bound than the optimistic residual. To do this, the relevance of an unjudged citation is set according to its probability of being relevant. To estimate this probability, we use the maximum likelihood estimation computed on the set of judged documents:

$$P(\texttt{relevant}|d) = \frac{|\texttt{relevant}|}{|\texttt{relevant}| + |\texttt{non-relevant}|}$$

This could be considered as a rough approximation of the number of relevant and non-relevant citations that would exist, if the relevance assessments for a query were complete. Intuitively, it provides a balance between the existing pessimistic view of unjudged citations and the aforementioned optimistic view of unjudged citations.

4 EXPERIMENTAL SETUP

Experiments were performed on a test collection of 125 systematic reviews [26]. The CLEF2017 TAR collection [8] was not used because that collection contains a smaller number of topics (50 in total) and therefore less training data. To perform the experiments,

Feature	Description	Applicability
Depth	How deep in the query the transformation was made.	Both
Clause Type	The type of clause the transformation was applied to (atomic/non-atomic).	Both
Children Count	Number of sub-clauses a non-atomic clause has	Non-Atomic
Transformation Type	The type of transformation (Table 1) made.	Both
Logical Operator Replacement Type	The type of operator this type of transformation modified.	Non-Atomic
MeSH Depth	The depth that the MeSH term was exploded from in the MeSH ontology.	Atomic
MeSH Parent	Whether a MeSH parent transformation is made.	Atomic
Restriction Type	Which type of field restriction was made when this transformation was applied.	Atomic
Clause Removal	Whether a clause removal transformation has been made.	Atomic
cui2vec	Whether a cui2vec Expansion has been made.	Atomic
cui2vec Expansions	The number of expansions that were made when this transformation was applied.	Atomic
MeSH Exploded	If the MeSH heading in an atomic clause is exploded or not.	Atomic
Truncated	If an atomic clause has a character replacement expression (e.g. wildcard).	Atomic
Retrieval Size†	Total number of citations retrieved by the query	Both
# Boolean Clauses†	Total number of Boolean clauses in the query.	Non-Atomic
# Boolean Keywords†	Total number of atomic clauses (keywords) in the query	Non-Atomic
# Truncated†	Total number of atomic clauses that have a truncated term.	Non-Atomic
# Fields†	Total number of fields in the query.	Both
# MeSH Keywords†	Total number of non-atomic clauses that contain a MeSH term.	Non-Atomic
# MeSH Exploded†	Total number of non-atomic clauses that have an exploded MeSH term.	Non-Atomic
# MeSH Non-Exploded†	Total number of non-atomic clauses that do not have an exploded MeSH term.	Non-Atomic
Average MeSH Depth†	The average depth MeSH terms in the query appear.	Non-Atomic
Maximum MeSH Depth†	The maximum depth MeSH terms in the query appear.	Non-Atomic
Δ	The difference of a feature between q and $\hat{q}$, where applicable (†).	Both
ω_i	n additional features that correspond to the sequence of transformations made.	Both

Table 2: Features used in the candidate selection task. Applicability denotes to which types of clauses a feature is applicable to: *Atomic* refers to clauses which consist of a single keyword, *Non-Atomic* refers to clauses which group several atomic clauses with a logical operator, and *Both* indicates that the feature can be applied to either type of clause.

we used a framework to construct pipelines for each stage of the experiments [23]. These pipelines and the code for the experiments are made available online.[3]

4.1 Query Candidate Generation

In order to train a model to select candidate queries, training data is required. We achieve this by exploring the space of queries using the transformation processes described in Section 3.2 up to a depth of 5. The difference here, however, is that rather than choosing one query as a candidate query, all queries are chosen. That is, rather than choosing one query to transition to in the chain, we explore all possible transitions in order to generate training data. Because this process is exponential in nature, we sample using the method described in Section 3.5. In terms of training statistics, on average 7968 queries were generated from a seed query. The highest number of queries generated for a topic was 129,282 and the lowest was 70. The total number of training data points was 717,160.

In the testing phase, where candidates for unseen queries are selected, the transformation process is the same as described in Section 3.2.

[3] Anonymised GitHub repository

4.2 Query Candidate Selection

The topics are split into approximately 70% train (90 topics) and 30% test (35 topics). In the training process, 30% of the train set is used for validation. To perform automatic candidate selection, we train two methods: a learning to rank model and a nearest neighbour approach. The queries that are refined by the candidate selection process are evaluated with six information retrieval measures (precision, recall, $F_{0.5}$, F_1, F_3 and work saved over sampling (WSS)). These measures are typically used to evaluate retrieval in the case of systematic reviews [8, 20, 26]. As the candidate selection function seeks to maximise an evaluation measure, a model is trained for each of the evaluation measures considered in this work for both the learning to rank and nearest neighbour candidate selectors, for a total of 12 models.

4.2.1 Learning to Rank. The DART learning to rank model is trained within the QuickRank framework [2]. We parametrise the DART model with a tree size of 100, a drop-out rate of 0.8, with uniform sampling, tree normalisation, a learning rate (shrinkage) of 0.1, and to optimise the ranking of queries using DCG@1.[4] We perform candidate selection using this model with the same ranking

[4] The exact parameters used to train the model can be found in the experiment pipeline files.

effectiveness metric (DCG@1). The choice of DCG@1 is motivated by the task of identifying the top-1 ranked candidate query to transition to in a query chain. We sample the training data by removing a large portion of negative examples (two-thirds) to train a DART learning to rank model, as we discovered empirically that negative examples degrade the ranking quality.

4.2.2 Nearest Neighbour. Our nearest neighbour approach first records the feature vectors and evaluation measure values for each query in the training set. Next, for a new unseen query, an evaluation measure value is approximated by searching for a query in the training set which minimises the distances between the two feature vectors and maximises the recorded evaluation measure score. The intuition here is to find the most effective query from the training set which is most similar to an unseen query. In order to find the top-1 candidate to continue the query chain, each generated candidate query is ranked by the approximated evaluation measure value described previously. For efficiency reasons, we only store the deltas of distance and score from the most effective query generated from each seed query. The entire training data is used to create a nearest neighbour model, as we empirically discovered that unlike learning to rank, with the nearest neighbour approach negative examples improved the ranking quality.

5 RESULTS AND ANALYSIS

The main results of the experiments are reported in Table 3, where statistical significant differences are computed using a paired two-tailed t-test with $p < 0.01$. Along with the results obtained when unjudged citations were considered not relevant, we also report results obtained by considering the optimistic and probabilistic (maximum likelihood estimation) residuals: these are indicated by the subscripts *r* and *mle*, respectively.

Overall, the results in Table 3 indicate that the automatic candidate selection methods identified refined queries that were better than the original query (Seed), for the chosen evaluation measures. Somewhat surprisingly for this task, the nearest neighbour candidate selector often outperformed the learning to rank selector, choosing candidate queries that more often refined the query, rather than degrading or broadening it. Notably, the nearest neighbour candidate selector that was trained to maximise the F_1 measure (indicated as $F1_n$ in Table 3) was the most effective at selecting queries that improve over the original seed queries in terms of precision, F_1, and F_3.

The summary of the effect each candidate selector has on the refined queries is presented in Table 4. These results suggest that, although the number of transformations that are made to queries is relatively small (sometimes less than five), the queries often differ significantly from the original seed queries. The final clauses included in the queries were impacted by the types of transformations that were allowed. The following mapping indicates which of the six transformations align with which headings in Table 4:

- Clauses, Keywords: *Clause Removal & cui2vec Expansion*
- # AND, # OR: *Logical Operator Replacement* (# NOT included for completeness)
- Fields, (Title, Abstract, MeSH, Other): *Field Restrictions*
- Exp. Mesh: *MeSH Explosion*
- Avg. MeSH, Max. MeSH: *MeSH Parents*

The query expansion method *cui2vec* used was not chosen often by the candidate selectors: on average, the selected queries did not contain more non-atomic clauses than the original seed queries (this is explored in more detail in Section 5.1).

To provide further insight into how each candidate selector model affects the resulting selected queries, Table 5 presents the average proportion of fields and Boolean operators in queries. As the number of clauses in each of the selected queries differ from each trained candidate selector, this table seeks to show the changes in the types of fields and types of Boolean operators in proportion to the size of the query (i.e., with respect to the total number of Boolean operators and fields). For example, the $F1_n$ selector (which consistently improved the effectiveness of queries over the original seed queries for all measures) tended to increase the number of Title fields while decreasing the number of Abstract fields, as well as increase the number of AND operators while decreasing the number of OR operators. These refinements to queries align with the intuition that using the Title field will yield fewer results than the Abstract (or both fields combined), and that the AND operator is more restrictive than the OR operator (the intersection of retrieved citations as opposed to the union).

In terms of directly optimising precision and recall, we further analyse the P_n, and R_l selectors, as each recorded the highest increase in precision and recall, respectively. The P_n selector showed a similar proportion of fields to $F1_n$, however the proportion of AND operators was lower and the proportion of OR operators was higher. The R_l selector had fewer Title *and* Abstract fields, and it removed atomic clauses that did not contain MeSH fields (this is implied by the fact that no transformation allowed MeSH terms to be added, and that it was possible for any clause to be removed with the *Clause Removal* transformation). In addition, this selector had a higher number of OR operators than AND operators. Finally, Table 4 shows that R_l is the only selector to increase the average MeSH depth, suggesting that his selector often applied the *MeSH Parent* transformation.

Next, we explore at a topic level the differences between candidate selectors and their effects on evaluation. Figures 4 and 5 present the topic-by-topic effectiveness of the selectors with respect to the target evaluation measures. The differences between the precision candidate selectors (i.e. the learning to rank selector and the nearest neighbour selector) is noticeable. The P_l selector in fact chose many queries which degraded the effectiveness in terms of precision. On the other hand, the P_n selector had (larger) gains for a similar number of topics, but when compared to the P_l selector, it shows both a reduced number of queries for which a loss was recorded, and a reduced amount of loss, when losses happened.

On the other hand, the R_l and R_n selectors were more similar, and many of the same topics had improved effectiveness. The results of these figures visualise the intuition that refining a query to increase precision is more difficult than recall. To contrast the previous two figures, Figure 6 presents the effect the $F1_n$ selector had on each topic in terms of the six evaluation measures considered in this work. While many queries had improved precision and maintained or even increased recall (e.g., topic 83), approximately one third of queries had a reduction in recall. This suggests that there may not have been enough training examples for this selector to choose a candidate query that is likely to cause an increase in recall.

	P	P_{mle}	P_r	R	R_{mle}	R_r	F0.5	$F0.5_{mle}$	$F0.5_r$	F1	$F1_{mle}$	$F1_r$	F3	$F3_{mle}$	$F3_r$	WSS	WSS_{mle}	WSS_r
Seed	0.0211	0.0307	0.7344	0.8395	0.8635	0.9831	0.0255	0.0358	0.7353	0.0373	0.0493	0.7366	0.1107	0.1331	0.7384	0.8386	0.8626	0.9822
P_l	0.0068	0.0142	0.7100	0.6989	0.7406	0.9591	0.0084	0.0161	0.7104	0.0129	0.0214	0.7109	0.0468	0.0610	0.7117	0.6849	0.7267	0.9450
P_n	0.0207	0.0432	0.5909	0.6403*	0.6810*	0.8541	0.0248	0.0472	0.5908	0.0355	0.0591	0.5908	0.0977	0.1331	0.5912	0.6396*	0.6802*	0.8533
R_l	0.0115	0.0195	0.7948	0.8833	0.8978	0.9580	0.0139	0.0222	0.7954	0.0201	0.0292	0.7963	0.0573	0.0716	0.7976	0.7946	0.8091	0.8694*
R_n	0.0023	0.0088	**0.7985**	**0.8939**	**0.9111**	0.9865	0.0029	0.0094	**0.7986**	0.0045	0.0113	**0.7988**	0.0183*	0.0270*	0.7990	**0.8551**	**0.8723**	0.9476
$F0.5_l$	0.0113	0.0237	0.7083	0.7826	0.8046	0.9513	0.0139	0.0254	0.7081	0.0213	0.0333	0.7079	0.0793	0.1011	0.7079	0.7802	0.8022	0.9489
$F0.5_n$	0.0191	0.0225	0.8206	0.8245	0.8355	0.9690	0.0231	0.0271	0.8215	0.0336	0.0393	0.8228	0.0965	0.1118	0.8245	0.8125	0.8235	0.9570
$F1_l$	0.0206	0.0229	0.7623	0.8330	0.8503	0.9787	0.0248	0.0275	0.7631	0.0360	0.0396	0.7644	0.1007	0.1090	0.7661	0.8224	0.8397	0.9681
$F1_n$	**0.0416**	**0.0638**	0.6474	0.6726*	0.7103*	0.9188	0.0486	**0.0604**	0.6454	**0.0655**	**0.0795**	0.6430	**0.1472**	**0.1772**	0.6408	0.6719*	0.7095*	0.9181
$F3_l$	0.0344	0.0248	0.6501	0.6860	0.7104*	0.9129	**0.0355**	0.0297	0.6497	0.0436	0.0424	0.6493	0.1041	0.1147	0.6490	0.6847	0.7091*	0.9116
$F3_n$	0.0175	0.0312	0.7279	0.7850	0.8195	0.9583	0.0212	0.0339	0.7191	0.0308	0.0448	0.7155	0.0910	0.1173	0.7144	0.7838	0.8183	0.9570
WSS_l	0.0152	0.0199	0.7666	0.7590	0.7961	0.9881	0.0181	0.0237	0.7665	0.0257	0.0335	0.7663	0.0634	0.0812	**0.7661**	0.7471	0.7842	0.9762
WSS_n	0.0229	0.0277	0.7323	0.7955	0.8177	**0.9867**	0.0275	0.0331	0.7329	0.0397	0.0474	0.7337	0.1131	0.1326	0.7349	0.7916	0.8138	**0.9828**

Table 3: Comparison of the effectiveness of the original queries in the test set (Seed) and each of the candidate selectors. Reported are the average values across topics. Values in bold indicate the highest value in that column. Statistically significance (two-tailed t-test with $p < 0.01$) is indicated with $*$. Evaluation methods are abbreviated in the following manner: P→precision, R→recall, WSS→work saved over sampling. Residual evaluation can be identified with the $_r$ (optimistic) and $_{mle}$ (maximum likelihood) subscripts. The two candidate selection functions can be identified by the evaluation measure they optimise, and a corresponding letter: e.g., P_l corresponds to the learning to rank selector which maximised precision, and $F1_n$ corresponds to the nearest neighbour selector which maximised F1.

	Clauses	# AND	# OR	# NOT	Fields	Fields (Title)	Fields (Abstract)	Fields (MeSH)	Fields (Other)	Keywords	Exp. MeSH	Avg. MeSH	Max. MeSH
Seed	12.6000	4.7714	7.3714	0.4571	74.4286	27.8286	29.3143	14.3143	2.9714	41.1714	4.2286	3.1763	7.7143
P_l	-3.3429*	-1.3714*	-1.8857*	-0.0857	-23.9143*	-9.2571	-9.3143	-4.8857*	-0.4571	-12.8857*	-1.5143*	-0.2009	-0.5714
P_n	-0.1714	0.3714	-0.5429*	0.0000	-2.9429*	0.3714	-2.7429*	-0.2000	-0.3714	-0.3429	-0.5143*	-0.2425	-0.2857
R_l	-2.8286	-1.6286	-1.0857	-0.1143	-18.5429	-7.1714	-8.1429	-2.3714	-0.8571	-11.0000	-0.7143	0.3190	-0.3143
R_n	-1.0857*	-1.0286*	0.0000	-0.0571	-5.9714*	-2.5429*	-1.9429*	-1.0000*	-0.4857	-2.6286*	-0.2857	-0.0877	-0.2286
$F0.5_l$	-2.3429*	0.0000	-2.2286*	-0.1143	-13.4000*	-3.7143*	-4.9429*	-4.2571*	-0.4857	-7.2857*	-1.2571*	-0.4887	-1.4571*
$F0.5_n$	-0.3143*	-0.1429	-0.1429	-0.0286	-2.3143*	-0.2571	-1.4000*	-0.2857	-0.3714	-0.3143*	-0.4571*	-0.1620	-0.3429
$F1_l$	-4.4286*	-1.5143*	-2.6286*	-0.2857*	-21.7143*	-6.9143*	-7.5143*	-6.4857*	-0.8000	-11.8000*	-2.0571*	-1.6380*	-3.9143*
$F1_n$	-0.3143	0.5143*	-0.8286*	0.0000	-3.3429*	-0.5714	-2.0286*	-0.4000	-0.3429	-0.7714	-0.7714*	-0.2804	-0.3143
$F3_l$	-0.4857	0.2857	-0.7714*	0.0000	-6.6000	-2.6000	-2.9429	-0.6857	-0.3714	-2.6286	-0.2000	-0.2266	-0.3429
$F3_n$	0.0000	-0.0286	0.0286	0.0000	-1.0571	-0.3429	-0.3429	0.0000	-0.3714	0.0000	-0.8286*	-0.3331*	-0.2857
WSS_l	-2.2857*	-0.8571*	-1.3714*	-0.0571	-12.7429*	-5.0286*	-5.1143*	-2.2000*	-0.4000	-6.1143*	-0.4857*	-0.4084	-0.8857
WSS_n	-0.8571*	-0.2571	-0.5714*	-0.0286	-5.9714*	-2.1143*	-2.4000*	-1.0000*	-0.4571	-2.8000*	-0.6571*	-0.1420	-0.3143

Table 4: Summary of the query refinement process. Shown in the table is the average difference in change of different aspects of queries obtained by each selector. A negative value indicates a reduction in the corresponding aspect, a positive value indicates an increase, and a zero value indicates that aspect was not changed. For comparison, the average values of the original *Seed* queries are reported as well. *Clauses* refers to the number of Boolean clauses in the query. *# AND, # OR* and *# NOT* indicate how many of the corresponding operators were in the query. *Fields* indicates how many fields were in the query in total; this is further divided into how many of those were Title, Abstract, MeSH, or other (e.g., dates or publication type). *Exp. MeSH* indicates the number of MeSH terms that have been exploded, and *Avg. MeSH* & *Max. MeSH* indicate the average and maximum depth in the ontology of the MeSH terms. Two-tailed statistical significance with $p < 0.01$ between the Seed queries and the queries selected by each model is indicated with $*$.

Finally, we compare two example queries by examining the transformations that have been applied to them. The queries chosen for comparison are from topics 73 and 154. These topics were respectively the best and worst performing ones when query transformations were obtained through the $F1_n$ candidate selector (in the case of topic 154 it was chosen because the query for 106 was too long to fit into a column). The transformed query for topic 73 obtained a large increase in precision, while maintaining recall, while the transformed query for topic 154 obtained a minor decrease in precision and a large decrease in recall.

```
1. measles.tw.
2. rubeola.tw.
3. morbilli*.tw.
4. exp Measles/
5. exp Measles virus/
6. or/1-5
7. exp Vitamin A/
8. vitamin a.ti,ab,sh.
9. retinol.ti,ab,sh.
10. exp Dietary Supplements/
11. or/7-10
12. bitot*.tw.
13. spot*.tw.
14. 12 and 13
15. vision*.tw.
16. visual*.tw.
17. eye*.tw.
18. sight*.tw.
19. or/15-18
20. xerosis*.tw.
21. exp Vision Disorders/
22. Xerophthalmia/
23. Night Blindness/
24. xerophthalmia*.tw.
25. exp Blindness/
26. keratomalacia.tw.
27. blind*.tw.
28. 14 or 19 or 20 or 21 or 22 or \
23 or 24 or 25 or 26 or 27
29. 6 and 11 and 28
```

(a) Original query.

```
1. measles.ti.
2. rubeola.tw.
3. morbilli*.ti.
4. exp Measles/
5. exp Measles virus/
6. or/1-5
7. Retinoids/
8. vitamin a.ti,ab,sh.
9. retinol.ti,ab,sh.
10. exp Dietary Supplements/
11. or/7-10
12. bitot*.tw.
13. spot*.tw.
14. 12 and 13
15. vision*.tw.
16. visual*.tw.
17. eye*.tw.
18. sight*.ab.
19. or/15-18
20. xerosis*.tw.
21. exp Vision Disorders/
22. Xerophthalmia/
23. Night Blindness/
24. xerophthalmia*.tw.
25. exp Blindness/
26. keratomalacia.tw.
27. blind*.ti.
28. 14 or 19 or 20 or 21 or 22 or \
23 or 24 or 25 or 26 or 27
29. 6 and 11 and 28
```

(b) Transformed query.

Figure 2: Comparison between the original query (left) and a transformed query (right) for topic 154. The transformed query was chosen using the $F1_n$ selector, and was the least effectively refined query. Note that on line 28 for both queries there is a carriage return as the line was too long.

The original and the transformed query for topic 73 are shown in Figure 3. The transformations made to the query were that lines 59 and 75 were both removed, and lines 40, 46 and 75 (which become line 66 in the transformed query) had their operators changed (from OR to AND). The original and the transformed query for topic 154 are shown in Figure 2. The transformations made to its query were that lines 1, 3, and 27 had their fields changed from Title and Abstract to only Title, and the MeSH term of line 7 was expanded to its parent.

Upon examination of the queries in Figures 3 and 2, it is clear that they considerably differ in length: one may question whether the length difference (in number of clauses) is associated with the effectiveness difference — according to this hypothesis shorter queries would generally perform worse than longer queries. However, upon further analysis we found that this is not the case. In fact, the Pearson's r correlation between the F_1 evaluation measure and the number of Boolean clauses in each query was only $r = 0.175$. Similar results were obtained when considering the other evaluation measures used in this study, thus suggesting that there is no direct relation between query length in terms of number of Boolean clauses and retrieval effectiveness. This finding is similar to what was obtained by Scells et al. when studying query performance predictors for this domain [22].

The next two sections explore our research questions with respect to the impact of semantic transformations in the candidate selection process, and the effect unjudged citations had on the evaluation.

5.1 Impact of Semantic Transformations

To answer **RQ1**: '*Which types of transformations are the most effective at refining Boolean queries in systematic review literature search?*', a comparison between candidate selection transformations was made. The summary of query refinement are presented in Table 4. On average, none of the approaches selected queries which performed explicit query expansion. Indeed, a deeper analysis of the results identifies that none of the queries are longer than the original seed. This suggests that, in this context, query expansion may not be an effective way to refine a query. On the

```
1. controlled clinical trial.pt.
2. placebo.ab.
3. clinical trials as topic/
4. randomly.ab.
5. trial.ti.
6. randomized.tw.
7. randomized controlled trial.pt.
8. or/1-7
9. humans/
10. animals/
11. 9 not 10
12. 8 and 11
13. abdominal.tw.
14. abdomen.tw.
15. chest.tw.
16. thoracic.tw.
17. trunk.tw.
18. or/13-17
19. exp Wounds, Penetrating/
20. 18 and 19
21. abdominal.tw.
22. abdomen.tw.
23. chest.tw.
24. thoracic.tw.
25. trunk.tw.
26. or/21-25
27. trauma*.tw.
28. injur*.tw.
29. penetrat*.tw.
30. stab*.tw.
31. or/27-30
32. 26 and 31
33. Splenic.tw.
34. spleen.tw.
35. stomach.tw.
36. gastric.tw.
37. or/33-36
38. rupture*.tw.
39. burst*.tw.
40. 38 or 39
41. 37 and 40
42. heart.tw.
43. cardiac.tw.
44. aortic.tw.
45. aorta*.tw.
46. or/42-45
47. rupture*.tw.
48. 46 and 47
49. exp Abdominal Injuries/
50. exp thoracic injuries/
51. 20 or 32 or 41 or 48 or 49 or 50
52. Blood.tw.
53. Plasma.tw.
54. 52 or 53
55. Autologous.tw.
56. 54 and 55
57. Autohaemotransfusion*.tw.
58. Auto-haemotransfusion*.tw.
59. Autohemotransfusion*.tw.
60. Auto-hemotransfusion*.tw.
61. Autotransfusion*.tw.
62. Auto-transfusion*.tw.
63. or/56-62
64. cell*.tw.
65. blood.tw.
66. 64 or 65
67. transfusion*.tw.
68. salvage.tw.
69. save*.tw.
70. or/67-69
71. 66 adj5 70
72. exp Blood Transfusion, Autologous/
73. exp Blood Loss, Surgical/
74. exp Blood Transfusion/
75. 63 or 71 or 72 or 73 or 74
76. 51 and 75
77. 12 and 76
```

(a) Original query.

```
1. controlled clinical trial.pt.
2. placebo.ab.
3. clinical trials as topic/
4. randomly.ab.
5. trial.ti.
6. randomized.tw.
7. randomized controlled trial.pt.
8. or/1-7
9. humans/
10. animals/
11. 9 not 10
12. 8 and 11
13. abdominal.tw.
14. abdomen.tw.
15. chest.tw.
16. thoracic.tw.
17. trunk.tw.
18. or/13-17
19. exp Wounds, Penetrating/
20. 18 and 19
21. abdominal.tw.
22. abdomen.tw.
23. chest.tw.
24. thoracic.tw.
25. trunk.tw.
26. or/21-25
27. trauma*.tw.
28. injur*.tw.
29. penetrat*.tw.
30. stab*.tw.
31. or/27-30
32. 26 and 31
33. Splenic.tw.
34. spleen.tw.
35. stomach.tw.
36. gastric.tw.
37. or/33-36
38. rupture*.tw.
39. burst*.tw.
40. 38 and 39
41. 37 and 40
42. heart.tw.
43. cardiac.tw.
44. aortic.tw.
45. aorta*.tw.
46. and/42-45
47. rupture*.tw.
48. 46 and 47
49. exp Abdominal Injuries/
50. exp thoracic injuries/
51. 20 or 32 or 41 or 48 or 49 or 50
52. Blood.tw.
53. Plasma.tw.
54. 52 or 53
55. Autologous.tw.
56. 54 and 55
57. Autohaemotransfusion*.tw.
58. Auto-haemotransfusion*.tw.
59. Auto-hemotransfusion*.tw.
60. Autotransfusion*.tw.
61. Auto-transfusion*.tw.
62. or/56-61
63. exp Blood Transfusion, Autologous/
64. exp Blood Loss, Surgical/
65. exp Blood Transfusion/
66. and/63-65
67. 51 and 66
68. 12 and 67
```

(b) Transformed query.

Figure 3: Comparison between the original query (left) and a transformed query (right) for topic 73. The transformed query was chosen using the $F1_n$ selector, and was the most effectively refined query.

	Title	Abstract	MeSH	Other	AND	OR	NOT	Exp. MeSH
Seed	0.3327	0.3589	0.2711	0.0374	0.3542	0.6172	0.0286	0.1227
$F1_l$	0.3366*	0.3764*	0.2324*	0.0547	0.3589*	0.6298*	0.0113*	0.0704*
$F1_n$	0.3432	0.3370*	0.2901	0.0297	0.4480*	0.5223*	0.0297	0.0819*
$F3_l$	0.3730	0.3500	0.2472	0.0298	0.4287	0.5417*	0.0296	0.1196
$F3_n$	0.3376	0.3706	0.2625	0.0293	0.3447	0.6267	0.0286	0.0819*
$F0.5_l$	0.3513*	0.3675*	0.2402*	0.0410	0.4406	0.5392*	0.0202	0.1048*
$F0.5_n$	0.3590	0.3501*	0.2621	0.0289	0.3387	0.6359	0.0254	0.1044*
P_l	0.3326	0.3657	0.2656*	0.0360	0.3246*	0.6402*	0.0352	0.1024*
P_n	0.3626	0.3198*	0.2882	0.0294	0.4263	0.5441*	0.0296	0.0936*
R_l	0.3161	0.3357	0.3221	0.0262	0.3196	0.6567	0.0236	0.1153
R_n	0.3252*	0.3822*	0.2647*	0.0280	0.2729*	0.6992	0.0278	0.1223
WSS_l	0.3378*	0.3559*	0.2771*	0.0293	0.3102*	0.6641*	0.0256	0.1358*
WSS_n	0.3314*	0.3504*	0.2899*	0.0283	0.3790	0.5927*	0.0283	0.1049*

Table 5: Average proportion of clauses for each field, each Boolean operator, and each Exploded MeSH term, across selected queries by each query selector approach. The original queries (Seed) are included for comparison. Statistically significance (two-tailed t-test with $p < 0.01$) between the *Seed* and listed selectors is indicated with $*$.

other hand, query reduction was chosen very often, more so by the learning to rank selector — which tended to remove clauses from queries more often than the nearest neighbour selector. While the candidate queries that had the *cui2vec Expansion* transformation applied were not chosen by either of the selectors, the *MeSH Parent* transformation was. The R_l selector chose queries with an average MeSH depth of 0.32 higher than the original seed query. This suggests that to maximise an increase in recall, this selector opted to chose a query that broadened the scope of MeSH keyword than to add new, somewhat related text-matching keywords. Overall the results indicate that, among the investigated transformations, *syntactic* transformations play a larger role in query refinement than *semantic* transformations.

5.2 Impact of Unjudged Citations

To answer **RQ2**: '*What effect do unjudged citations have on the effectiveness of refined Boolean queries in systematic review literature search?*', the evaluation measures and the estimates obtained with the two residual heuristics are compared. As reported in Table 3, these results show that the effect of unjudged citations is not significant. When a selector chooses candidate queries that perform statistically significantly worse for an evaluation measure, the residual *mle* results are also significantly worse. Nonetheless, Table 3 still shows that in many cases, effectiveness increases when unjudged citations are factored into evaluation. For example, the P_n selector does not obtain increases in precision over the seed queries, when unjudged citations are considered non-relevant. However, when using the *mle* heuristic, precision is higher, though with the *r* heuristic, precision is also lower. In fact, from analysing the *r* heuristic, we can determine that the P_n selector retrieves many

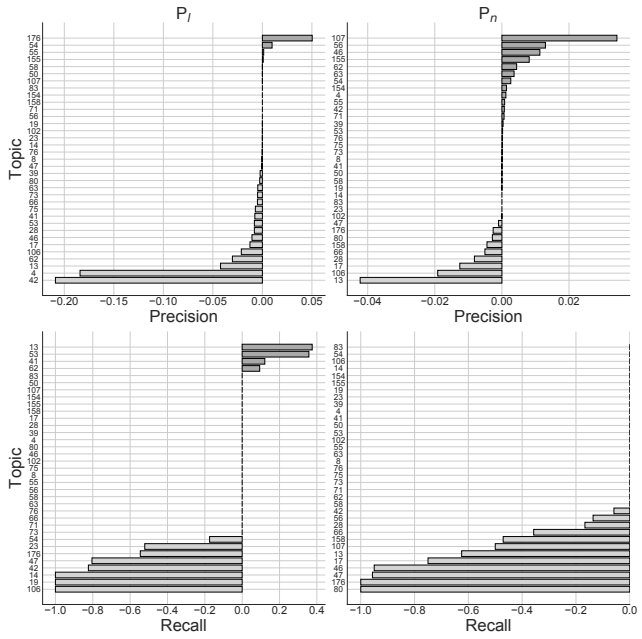

Figure 4: Per-topic gains in precision (top) and recall (bottom) over the Seed query for the candidate selectors which maximised precision (P_l and P_n).

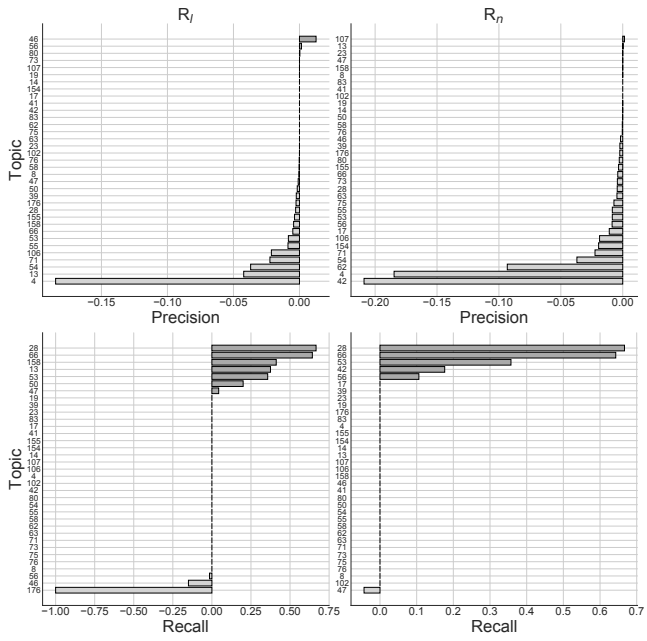

Figure 5: Per-topic gains in precision (top) and recall (bottom) over the Seed query for the candidate selectors which maximised recall (R_l and R_n).

more non-relevant citations than the P_l selector. For these reasons, estimates for residual should be considered when performing high-recall tasks such as this; or in tasks that modify or change

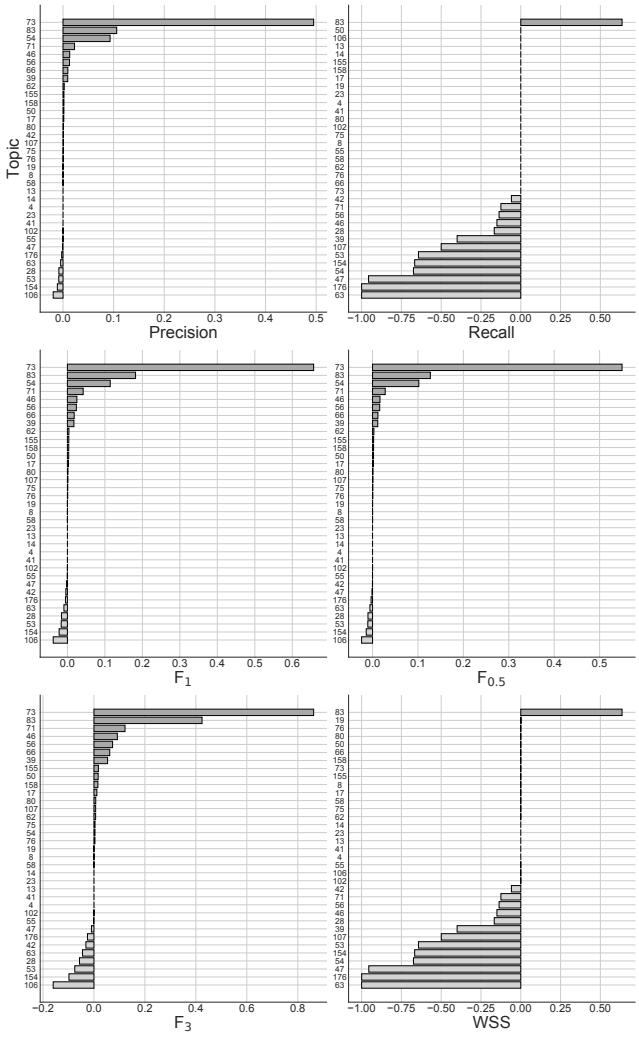

Figure 6: Per-topic gains over the Seed query in the evaluation measures considered in our study. Here we considered the nearest neighbour candidate selector that maximised F1 ($F1_n$.)

queries in some way. This is especially important when relevance assessments are largely incomplete, and no system variation was considered when forming the pool for assessments: this was the case for systematic review collections available for Information Retrieval research [8, 26].

6 CONCLUSIONS

This paper presented three novel semantic transformations that integrate with previous work on systematic review query generation. In addition, we cast the task of query generation from previous work [24] to query refinement — where newly generated queries should improve precision while maintaining or improving recall. To this end, we employed two new candidate selector functions, one based on learning to rank, and one based on a nearest neighbour algorithm. Both candidate selectors were trained to maximise six evaluation measures. When evaluating query refinement, we also revealed that the refinement retrieved unjudged documents; we accounted for these in the evaluation by computing the residuals of the evaluation measures. We did so by considering both an optimistic residual and a probabilistic residual (based on maximum likelihood estimation). Our results showed that the nearest neighbour model was able to more effectively select query refinements than the learning ro rank approach. Our experiments have shown that the explicit query expansion method (*cui2vec Expansion*) proposed in this work is not viable for query refinement (as candidates with other transformations outrank it). However, the explicit query reduction method (*Clause Removal*), and to a lesser extent the implicit query expansion method (*MeSH Parents*) can, in fact, be used to help refine queries.

The detailed analysis of how each set of selected queries was transformed by candidate selectors suggested that the syntactic transformations were in fact more suited to refinement than semantic transformations. Our experiments also showed that the effect of unjudged citations on evaluation is significant: unjudged citations should be accounted for in some way when performing evaluation.

Future plans for the task of automatic query refinement for systematic review literature search will focus on exploring other methods for candidate selection and other features, as well as improving the sampling of queries in the exploration/generation phase for collecting training data. With this in mind, the current sampling method leads to a large imbalance of negative to positive training samples. One promising sampling method could be to sample each query with a set of "known-relevant" seed documents. These are used in systematic review literature search by information specialists to guide the query generation process. Recently, a seed-driven approach to ranking has been shown to reduce the workload of the screening task [15]. We also plan to further study other techniques for semantic query transformation, such as query reduction [13]. Query reduction techniques for professional search have been shown to be successful in other contexts, both in the clinical domain [11, 12, 28], and outside of the medical field [4, 16].

The proposed method for automatic query refinement has the potential to vastly reduce the time spent creating systematic reviews. The searching and screening processes contribute to a significant period of time and money in the systematic review creation process. By reducing the total number of citations to screen by automatically refining already highly effective queries, the time spent screening citations and therefore the total cost of a systematic review can be reduced. These reductions lead to faster and more up-to-date evidence based medicine which in turn lead to more accurate clinical decisions and better health outcomes.

Acknowledgements. These will be included in the final version. The authors require 3 lines of text for the acknowledgements, as shown here.

REFERENCES

[1] Andrew L. Beam, Benjamin Kompa, Inbar Fried, Nathan P. Palmer, Xu Shi, Tianxi Cai, and Isaac S. Kohane. 2018. Clinical Concept Embeddings Learned from Massive Sources of Medical Data. *CoRR* abs/1804.01486 (2018). arXiv:1804.01486 http://arxiv.org/abs/1804.01486

[2] Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. 2016. Quality versus efficiency in document scoring with learning-to-rank models. *Information Processing & Management* 52, 6 (2016), 1161–1177.

[3] Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)* 44, 1 (2012), 1.

[4] Debasis Ganguly, Johannes Leveling, Walid Magdy, and Gareth JF Jones. 2011. Patent query reduction using pseudo relevance feedback. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 1953–1956.

[5] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. 2008. A survey of pre-retrieval query performance predictors. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 1419–1420.

[6] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 41–48.

[7] Jimmy, Guido Zuccon, and Bevan Koopman. 2018. Choices in Knowledge-Base Retrieval for Consumer Health Search. In *Advances in Information Retrieval*, Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury (Eds.). Springer International Publishing, Cham, 72–85.

[8] E. Kanoulas, D. Li, L. Azzopardi, and R. Spijker. 2017. CLEF 2017 Technologically Assisted Reviews in Empirical Medicine Overview. In *CLEF'17*.

[9] S. Karimi, S. Pohl, F. Scholer, L. Cavedon, and J. Zobel. 2010. Boolean versus ranked querying for biomedical systematic reviews. *BMC MIDM* 10, 1 (2010), 1.

[10] Y. Kim, J. Seo, and W. B. Croft. 2011. Automatic boolean query suggestion for professional search. In *SIGIR'11*.

[11] Bevan Koopman, Peter Bruza, Guido Zuccon, Michael Lawley, and Laurianne Sitbon. 2012. Graph-based Concept Weighting for Medical Information Retrieval. In *Seventeenth Australasian Document Computing Symposium (ADCS 2012)*. Dunedin, New Zealand.

[12] Bevan Koopman, Liam Cripwell, and Guido Zuccon. 2017. Generating clinical queries from patient narratives: A comparison between machines and humans. In *Proceedings of the 40th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 853–856.

[13] Giridhar Kumaran and Vitor R Carvalho. 2009. Reducing long queries using query quality predictors. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 564–571.

[14] J. Lavis, H. Davies, A. Oxman, J.L. Denis, K. Golden-Biddle, and E. Ferlie. 2005. Towards systematic reviews that inform health care management and policy-making. *JHSRP* 10, 1_suppl (2005), 35–48.

[15] Grace E. Lee and Aixin Sun. 2018. Seed-driven Document Ranking for Systematic Reviews in Evidence-Based Medicine. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. ACM, New York, NY, USA, 455–464. https://doi.org/10.1145/3209978.3209994

[16] Daniel Locke, Guido Zuccon, and Harrisen Scells. 2017. Automatic Query Generation from Legal Texts for Case Law Retrieval. In *Asia Information Retrieval Symposium*. Springer, 181–193.

[17] Jessie McGowan and Margaret Sampson. 2005. Systematic reviews need systematic searchers (IRP). *Journal of the Medical Library Association* 93, 1 (2005), 74.

[18] Mandar Mitra, Amit Singhal, and Chris Buckley. 1998. Improving automatic query expansion. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 206–214.

[19] Alistair Moffat and Justin Zobel. 2008. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)* 27, 1 (2008), 2.

[20] Alison O'Mara-Eves, James Thomas, John McNaught, Makoto Miwa, and Sophia Ananiadou. 2015. Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Systematic reviews* 4, 1 (2015), 5.

[21] KV Rashmi and Ran Gilad-Bachrach. 2015. Dart: Dropouts meet multiple additive regression trees. In *International Conference on Artificial Intelligence and Statistics*. 489–497.

[22] H. Scells, L. Azzopardi, G. Zuccon, and B. Koopman. 2018. Query Variation Performance Prediction for Systematic Reviews. In *SIGIR'18*.

[23] H. Scells, D. Locke, and G. Zuccon. 2018. An Information Retrieval Experiment Framework for Domain Specific Applications. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

[24] H Scells and G Zuccon. 2018. Generating Better Queries for Systematic Reviews. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. ACM, New York, NY, USA, 475–484.

[25] H. Scells, G. Zuccon, B. Koopman, A. Deacon, L. Azzopardi, and S. Geva. 2017. Integrating the framing of clinical questions via PICO into the retrieval of medical literature for systematic reviews. In *CIKM'17*.

[26] H. Scells, G. Zuccon, B. Koopman, A. Deacon, S. Geva, and L. Azzopardi. 2017. A Test Collection for Evaluating Retrieval of Studies for Inclusion in Systematic Reviews. In *SIGIR'2017*.

[27] L. Soldaini, A. Cohan, A. Yates, N. Goharian, and O. Frieder. 2015. Retrieving medical literature for clinical decision support. In *ECIR'15*. 538–549.

[28] Luca Soldaini, Andrew Yates, and Nazli Goharian. 2017. Learning to reformulate long queries for clinical decision support. *Journal of the Association for Information Science and Technology* 68, 11 (2017), 2602–2619.

[29] G. Tsafnat, P. Glasziou, M.K. Choong, A. Dunn, F. Galgani, and E. Coiera. 2014. Systematic review automation technologies. *SR* 3, 1 (2014), 74.

[30] 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*. ACM, 4–11.