2018

A Wikipedia powered state-based approach to automatic search query enhancement

Kyle Goslin

Markus Hofmann

A Wikipedia powered state-based approach to automatic search query enhancement

Article *in* Information Processing and Management · November 2017

DOI: 10.1016/j.ipm.2017.10.001

2 authors:

Kyle Goslin
Technological University Dublin - Blanchardstown Campus
22 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Markus Hofmann
Technological University Dublin - Blanchardstown Campus
81 PUBLICATIONS **352** CITATIONS
SEE PROFILE

A Wikipedia Powered State-based Approach to Automatic Search Query Enhancement

Kyle Goslin[1]

Institute of Technology Blanchardstown

Markus Hofmann[2]

Institute of Technology Blanchardstown

Abstract

This paper describes the development and testing of a novel Automatic Search Query Enhancement (ASQE) algorithm, the Wikipedia N Sub-state Algorithm (WNSSA), which utilises Wikipedia as the sole data source for prior knowledge. This algorithm is built upon the concept of iterative states and sub-states, harnessing the power of Wikipedia's data set and link information to identify and utilise reoccurring terms to aid term selection and weighting during enhancement. This algorithm is designed to prevent query drift by making callbacks to the user's original search intent by persisting the original query between internal states with additional selected enhancement terms. The developed algorithm has shown to improve both short and long queries by providing a better understanding of the query and available data. The proposed algorithm was compared against five existing ASQE algorithms that utilise Wikipedia as the sole data source, showing an average Mean Average Precision (MAP) improvement of 0.273 over the tested existing ASQE algorithms.

Keywords: Automatic Search Query Enhancement, Query Drift, Information Retrieval, Wikipedia

1. Introduction

The process searching for content on the web is typically done by entering search terms into a text field on the front-end of a search engine. This process however, can be seen as a one-size-fits-all approach, generalising the user's requirements, needs, background knowledge with search engines and overall search ability. Automatic Search Query Enhancement (ASQE) algorithms aim to enhance user submitted queries but often require additional information that can be used as a source of candidate expansion terms and as a method to gauge the importance of available terms. Recent ASQE algorithms (Bruce et al. 2012; ALMasri et al. 2013; Boston et al. 2014; Zhao et al. 2014; Zingla et al. 2016) have shown the use of dynamic data sources, such as Wikipedia[3], which offers high quality and ever changing articles with common fields and structure, can be beneficial to the ASQE process. Many ASQE approaches however, become so focused on the term identification process, they do not consider the true relevance the terms have to the user's query as a whole. It

[1]Corresponding author: E-mail kyle@cct.ie; Department of Informatics and Engineering, Blanchardstown Road North, Dublin 15, Ireland, Ireland

[2]Department of Informatics and Engineering, Blanchardstown Road North, Dublin 15, Ireland

[3]https://www.wikipedia.org

can often be the case that too many enhancement terms are added or that the terms that have been selected are of questionable relevance or conceptually distant causing *query drift* (Shtok et al., 2012).

The objective of this research is to utilise all available data from the Wikipedia data set and term relevance metrics for each data source to automatically enhance search queries to improve the precision of search results. The theory behind this research is that a state based approach provides a number of chances for reoccurring terms be identified, and when an alternative method for selection of available candidate terms is used, it will allow optimal candidate terms to surface. In addition to this, during the process of gathering of related data for a given query, if calls are made back to the original query to provide constant relevance persistence of the user's original intent with selected candidate terms, the collection of enhancement terms will be more relevant to the user.

To do this, a novel state and sub-state based approach to ASQE was developed with a stem query and a Term Window based selection process of candidate terms. The proposed algorithm, the Wikipedia N Sub-state Algorithm (WNSSA) is described and tested using 50 of the TREC-9 & 50 TREC 2014 Web Topics[4] on the ClueWeb12 full data set[5]. In addition to this, five existing ASQE algorithms that utilise different aspects of the Wikipedia data set were implemented and analysed. Relevance calculations for each algorithm are performed using the Average Precision@10 results from each of the enhanced sample topics and the overall Mean Average Precision@10 for each tested algorithm.

The main contributions defined in this paper include: 1) A novel state based approach to the process of gathering and identifying candidate enhancement terms for ASQE; 2) Stem query generation and utilisation between states during the Information Retrieval (IR) process for ASQE, consisting of the user's original query and identified relevant enhancement terms from the current internal states to prevent query drift; 3) Term Window based selection of enhancement terms for ASQE; and 4) A comprehensive cross-analysis of five existing ASQE algorithms that utilise Wikipedia as the sole data source for candidate enhancement terms against the proposed algorithm. This paper begins in Section 2 reviewing the area of ASQE and query drift, with a focus on algorithm that utilise Wikipedia as the data source for expansion terms. Section 3 provides an overview of the methodology followed and Section 4 outlining the ASQE algorithms tested and the data sets utilised during the testing and analysis process. The core focus of this paper is the developed ASQE algorithm, the WNSSA, which is described in Section 5. To provide an understanding of the results, Section 6 discusses the results of the testing process of the five existing Wikipedia powered ASQE algorithms and the proposed algorithm. Section 7 concludes this research outlining the key findings. As work on the WNSSA is ongoing, Section 8 outlines the future work for this study.

2. Related Work

ASQE is the process of automatically enhancing a user search query, typically through the addition, removal or correction (Vilares et al., 2016) of search terms to improve precision / recall of a search query. Ongoing research (Zingla et al., 2016) has continued to show that Wikipedia is useful as a source of prior knowledge to aid ASQE algorithms due to the quantity and wide domain of topics available. Unlike utilising static document collections and thesauri[6] which require expert knowledge to maintain, Wikipedia has shown to be beneficial as a source of prior knowledge for

[4] http://trec.nist.gov/data/web_topics.html
[5] http://lemurproject.org/clueweb12/
[6] https://wordnet.princeton.edu/

domain specific query enhancement such as in the area of patent retrieval (Al-Shboul & Myaeng, 2014; Sharma et al., 2015).

As ASQE is built upon a number of different IR techniques, each component of the ASQE process can be further enhanced by utilising Wikipedia; such as term weighting (Karisani et al., 2016), linguistic understanding (Selvaretnam & Belkhatir, 2016), relevance calculation (Zhao & Callan, 2012), term disambiguation (Habibi et al., 2016; Yadav & Kumar, 2016) and similarity assessment based up Wikipedia articles (Jiang et al., 2015). The additional data available in Wikipedia can be beneficial to users, as during search, users with little knowledge about the area of search have shown to perform worse due to their lack of prior knowledge when compared to domain experts (Monchaux et al., 2015). He & Ounis (2009) identified two possible reasons for the failure of ASQE, low query quality and topic drift. As search queries can be overly simple or complex, recent research has moved towards understanding the structural and syntactic complexity of search queries (Roy et al., 2016), which can further improve ASQE techniques. The context of a user during search plays an important role as it often does not exist for the user at the beginning of a search session (Fourney & Dumais, 2016).

ALMasri et al. (2013) proposed a Wikipedia based semantic query enrichment algorithm, whereby semantically related terms are extracted from Wikipedia and then used as Pseudo Relevance Feedback (PRF). This process is achieved through the following steps: Collect all articles $S(q)$ which are entitled by the user's query q. Each article $a \in S(q)$ has the probability $P(a \mid q)$ of being used in the enrichment process. The probability is defined as $P(a \mid t) = \frac{|O(a)|}{\sum_{a_i \in S(t)} |O(a_i)|}$, where $O(a)$ is the set of articles that a points to. The expansion set ES of selected n number of articles for user query q are defined as $ES(q,n) = \bigcup_{a \in S(q)} f(a, \lceil n \times P(a \mid q) \rceil)$. The collection of terms for query q are built from a union of article titles in the enrichment set. A weight is attached to each between 0 and 1, whereby 1 is most important and 0 is least important. The weight for each of the terms is defined as $weight(t, q_e) = \alpha \times SIM(a_q, a_t)$, whereby α is a tuning parameter between 0 and 1. The similarity calculation between two articles, a_1 and a_2, is defined in Equation 1, where $I(a)$ is the set of articles that points to a.

$$SIM(a_1, a_2) = \frac{\mid I(a_1) \cap I(a_2) \mid + \mid O(a_1) \cap O(a_2) \mid}{\mid I(a_1) \cup O(a_1) \mid + \mid I(a2) \cup O(a2) \mid} \tag{1}$$

Boston et al. (2014) proposed a tool titled Wikimantic which exploits Wikipedia articles and their inter-article reference relations which has shown to be effective for short queries. They define an AtomicConcept as a simple form of a concept. Each article is considered a series of terms which was generated by an AtomicConcept. The prior probability of $P(A)$ generating terms, where A is an individual article is defined as $P(A) = \frac{number\ of\ incoming\ links}{number\ of\ links\ in\ Wikipedia}$. As most of the articles in Wikipedia are linking to other articles, the authors define the probability of article A generating term t is defined as $P(t \mid A) = \frac{count(t,A)}{number\ of\ words\ in\ A}$.

Due to the limitation that not all articles will have a variety of different terms to check their probability with, the Microsoft n-gram corpus[7] containing 100,000 unique terms is used. Building upon an AtomicConcept, a new variant is defined as a MixtureConcept which is a collection of different AtomicConcepts. A MixtureConcept is defined as $M = \{(w_i, A_i) \mid i = 1...n\}$, where w_i is the weight of the concept and A is an individual article concept. In this Equation, i is the current AtomicConcept being viewed inside of M and w_i is the weight of A_i in MixtureConcept M. The probability of a MixtureConcept, $P(M)$ generating terms is defined as $P(M) = \sum_{i=1}^{n} w_i * P(A_i)$. The probability of generating term t for mixture concept M is defined as $P(t \mid M) = \sum_{i=1}^{n} w_i * P(t \mid A_i)$.

[7] http://research.microsoft.com/apps/pubs/default.aspx?id=130762

After an AtomicConcept set has been generated, a weight w_i is applied. S is the number of terms in the Concept. This is shown in Equation 2 and the probability of $P(A)$ generating term t is shown in Equation 3.

$$w_i = P(A_i \mid S) = \prod_{j=1}^{|S|} P(A_i \mid t_j) \tag{2}$$

$$P(A_i \mid t_j) = \frac{P(t_j \mid A_i) * P(A_i)}{P(t_j)} \tag{3}$$

Given query Q, a set of MixtureConcepts are created and then Equation 4 is used for generating possible expansion terms where $P(t \mid A_i)$ is the likelihood of generating term t from the AtomicConcept A_i and w_i is the weight of the AtomicConcept A_i in MixturenConcept M for user submitted query Q described as $M(Q)$. N is the number of documents in the collection and $df(t)$ is the number of documents that contain t. $ln\frac{N+1}{df(t)}$ is the described as the IDF weighting for the given term t.

$$ExpWeight(t \mid M(Q)) = \sum_{A_i \in M(Q)} P(t \mid A_i) \times w_i \times ln\frac{N+1}{df(t)} \tag{4}$$

Xu et al. (2009) outlined a query dependant PRF approach based on Wikipedia. They first began with an approach to categorise user queries into three categories, entity queries, ambiguous queries, and broader queries. They also proposed a number of different approaches for enhancement including a Relevance Model based approach, Field Evidence approach utilising the fields identified in a Wikipedia page and an Entity Page based approach. Their results show that the Entity Page based approach was the most successful and is discussed below. Instead of focusing on the top ranked documents as shown above, this approach focuses on utilising the Entity Page, e.g., the page which corresponds directly to the topic that the user is searching as an initial source of additional terms for PRF. The procedure followed during this study is defined as: 1) Identify an entity page for the user submitted query Q, 2) All terms on the entity page are ranked using TF-IDF, and 3) Top k terms are extracted, where the score for a term t on an Entity Page is defined using TF-IDF, where tf is the TF on an entity page and idf is computed as $log(N/DF)$, and n is the number of documents in the Wikipedia collection and DF is the number of documents that contain term t defined as $score(t) = TF - IDF$.

Bruce et al. (2012) described query expansion powered by Wikipedia hyperlinks. This approach begins by first breaking a query into query aspects. Poorly represented areas of the query are then enhanced with additional terms. This process contains six steps outlined as: 1) The user's initial query is recieved, 2) aspects of the query are identified, 3) Wikipedia articles are selected, 4) aspect vocabulary is constructed, 5) finding under represented aspects, and 6) query expansion. For aspect identification, Link Probability Weighting is used. This is done by counting the number of documents where the term is already a hyperlink divided by the number of documents where the term appeared. Aspects are selected from the highest value through to the lowest. An aspect is ignored if it is a subset of an already selected aspect. No aspects with weighting of 0 should be added, unless they contain terms that are yet to be covered by selected aspects. Aspect Identification is complete when each term of the query has been covered by an aspect. A collection of articles is created with a connection to the aspects defined. Each of the aspects are disambiguated individually using Link Probability measure. A cut-off threshold is then utilised, and all articles that have a confidence greater than half of the maximum measure are added. Aspects are disambiguated into pairs using the Wikipedia Link Based measure, described in Equation 5. The similarity between two articles is

defined, where A and B are the set of articles that link to a and b and W is the entire Wikipedia collection.

$$sr(a,b) = \frac{log(max(|\ A\ |, |\ B\ |)) - log(|\ A \cap B\ |)}{log(|\ W\ |) - log(min(|\ A\ |, |\ B\ |))} \tag{5}$$

Aspect vocabulary construction aims to build a weighted vocabulary for each aspect using the article set created previously. All terms appearing in the selected articles are vocabulary candidates weighted by their relation to their corresponding aspects. Each candidate term is added along with the Wikipedia Link Based Measure score. Finally, under represented aspects are identified. This step selects the best expansion term by counting term frequencies of all terms in the first 10 documents of an initial Bing search. The first 50 highest weighted terms are normalized. Scores are then calculated by multiplying term weights in the aspect vocabulary by their frequency weighting in the query vocabulary. From this the lowest score is determined to be the least represented. The aspects vocabulary is assigned as the final output for query expansion.

Zhao et al. (2014) described a novel term semantic query model based on Wikipedia. This approach is focused upon finding the semantic relatedness between terms using Wikipedia. The semantic correlation of to terms is defined as $T_jW_i = TF_i * log(\frac{N_1+N2}{n})$, where i,j = 1,2 and T_jW_i represents the weight of the *ith* common words in T_j word groups. The summary paragraph available for all Wikipedia articles is used to compute the semantic relatedness of two terms shown in Equation 6, where a,b represents two terms that are used for semantic computing and T_i, T_2 represent the word group obtained by word segmentation on the summary paragraph, N_1, N_2 are the number of words in word group T_1, T_2.

$$sim_a(a,b) = \frac{MAX(N_1,N_2)}{MIN(N_1,N_2)} + \sum_{i=1}^{n} T_i w_i * T_2 W_i \tag{6}$$

Semantic link relatedness is computed using Equation 7, whereby a and b represent two terms that are used for semantic computing, A is the number of inbound links for term a and B is the number of inbound links for term b. W is defined as the number of individual articles in Wikipedia.

$$sim(a,b)_{in} = \frac{log(MAX(|\ A\ |, |\ B\ |)) - log(|\ A \cap B\ |)}{log(|\ W\ |) - log(\text{min}(|\ A\ |, |\ B\ |))} \tag{7}$$

Zhao et al. (2016) described a method for Named Entity Disambiguation, which contains a query expansion based upon the utilisation of Wikipedia terms based on co-occurrence mentions. The authors describe that often, in the case of an article, a name is mentioned in complete form at the start of an article. Two main strategies for identifying candidates are: 1) queries that contain abbreviations, a match is made to terms which have similar capitalisation; 2) queries that contain continuous strings where the first letter of the string is also a capital letter, a match can be made to a candidate. The Wikipedia data utilised by this approach includes article titles, article content and article redirections. In their method, an initial query is placed to collect the top-k documents for a given query. Any candidate terms that are identified in the article collection become part of the collection of enhancement terms and articles returned become part of the article collection. The authors of this method identify that their query expansion approach is simplistic, and titled it the feedback-query-expansion method, as it incorporates a feedback loop to find candidates during retrieval. Due to the simplistic nature of this approach, it will be excluded from the testing described in this paper.

Zingla et al. (2016) described the issue of short queries in microblog retrieval and implemented ASQE using Wikipedia. The authors' method of identify candidate expansion terms was done by, 1) selecting unstructured full texts related to the original query utilising TF-IDF to select similar texts,

2) texts are tagged using the TreeTagger, 3) extract nouns from text, and 4) generate association rules using the CHARM algorithm. After a collection of candidate terms has been generated, additional processing is performed to ensure the terms are related to the original query terms. This is done through the use of their proposed semantic relatedness measure titled ESAC which combines Explicit Semantic Analysis using Wikipedia and association rules' confidence measures. This is described in Equation 8 where $Conf_{max}(R, q, w)$ is the max of the confidence of any association rule from R_j from rule collection R. $ESA(q, w)$ is the score of relatedness between the query q and the candidate term w and α is a tuning parameter between 0 and 1.

$$ESAC(q, w) = \begin{cases} (\alpha \times ESA(q, w) + (1 - \alpha) \times Conf_{max}(R, q, w) if Conf_{max}(R, q, W) \neq 0; \\ ESA(q, w), otherwise. \end{cases} \quad (8)$$

After this process is completed, the most related terms are then added to the original search query. Their research showed that the best results were achieved with rule mining and a term filtering phase which used a Wikipedia-based ESAC to prevent similar terms being utilised in the enhanced query.

3. Methodology

In this research, the testing and analysis procedure followed during the enhancement and ranking of search topics for each existing algorithm, and the developed ASQE algorithm, proposed by this paper, are defined as:

1. Select test topic, Q, from test query collection.
2. Pass Q to the current enhancement algorithm under analysis.
3. Gather 10 generated terms from the selected enhancement algorithm.
4. Merge original query Q and the new additional enhancement terms.
5. Pass the enhanced query to the ClueWeb12 full data set Batch Query Service to retrieve results.
6. Calculate the Average Precision @10 for the given enhanced query based on the results returned.

The Average Precision @10 was calculated by first analysing the top ten results returned per enhanced query from the ClueWeb12[8] full data set Batch Query Service[9] running the Lemur IR engine[10]. The topic description was then gleaned from the description field for each topic from the given TREC-9[11] & TREC 2014[12] Web Topic collections. If the result returned was relevant, the result was marked with 1, if the result was irrelevant it was marked as 0. For each test completed, 500 manual relevance assessments were performed. In addition to the Average Precision @10, the Mean Average Precision for each test was also calculated providing an overall score for each algorithm tested.

[8] http://www.lemurproject.org/clueweb12.php/
[9] http://boston.lti.cs.cmu.edu/Services/clueweb12_batch/
[10] https://www.lemurproject.org/
[11] http://trec.nist.gov/data/topics_eng/topics.451-500.gz
[12] http://trec.nist.gov/data/web/2014/web2014.topics.txt

4. Experimentation

In this research, three collections of tests were performed. In the first collection of tests, five algorithms, each of which utilised components from Wikipedia article content, Wikipedia API, search functionality, collection statistics or link analysis techniques using the Wikipedia, were analysed. As no concrete algorithm existed for each algorithm, each was recreated using the Python 2.7 programming language based on the specification defined by the authors. As research in the area of ASQE algorithms based on Wikipedia is limited, out of the six algorithms outlined in Section 2, the Algorithm by Zingla et al. (2016) was omitted as it was based on Rule Mining, which was conceptually distant from the algorithm proposed in this paper. The five remaining algorithms were chosen for analysis. These algorithms include: Algorithm 1 by ALMasri et al. (2013) utilised Search API, Article Titles and Article Content; Algorithm 2 by Boston et al. (2014) utilised Inter-article Link References and Article Content; Algorithm 3 by Xu et al. (2009) utilised Article Content, Search API and Wikipedia Document Collection Size; Algorithm 4 by Bruce et al. (2012) utilising Inter-article link references and Article Content; Algorithm 5 by Zhao et al. (2014) utilised Search API, and Article Summary Text. Each of these algorithms were provided 50 of the TREC-9 Web Topics for enhancement. After the enhancement was performed, the resulting enhanced query was passed to the ClueWeb12 full data set Batch Query Service to retrieve documents for relevance assessment.

The second collection of tests focused on the Wikipedia N Sub-state Algorithm (WNSSA) which is proposed in this paper. To provide a relevant baseline, the same test topics and data set as those in the first collection of tests were utilised during these tests.

The third collection of tests contained 30 individual tests in which the proposed algorithm was tested with variable tuning parameters to identify the most useful to aid enhancement. For each of the 30 tests, 50 of the TREC 2014 Web Topics were utilised and tested on the ClueWeb12 full data set using the Batch Query Service. For this research, the TREC-9 & TREC 2014 Web Topics were chosen as they contain length variation, spelling issues, domain diversity and lexical variations. In the TREC-9 test topics, the average query length is 3.44 tokens long and standard deviation of 2.67 , and the TREC 2014 Web Topics contain an average length of 3.3 tokens and a standard deviation of 1.38. To query the ClueWeb12 full data set, the ClueWeb12 Batch Query Service was utilised which uses the query-likelihood model with Dirichlet smoothing. This service allows 50 queries to be passed and returned in one of two formats, trec_eval or Indri default format. For this research the Indri default format was utilised. The index utilised by the Batch Query Service was created using the IndriBuildIndex, the data was processed with the default stoplist[13], and stemmed using the Krovetz Stemmer [14].

Both topic collections contain sub-topics which describe each topic. During these three collection of tests, 18,000 individual manual relevance assessments were performed to ensure the sub-topics outlined could be seen in the result collection. For each enhancement algorithm analysed, 10 enhancement terms were added. This was based upon research by Ogilvie et al. (2009) which outlined that 10 or less provided the optimal enhancement. As this research is not focused on optimisation of this parameter, 10 enhancement terms was chosen for each tested algorithm.

[13] http://boston.lti.cs.cmu.edu/Services/clueweb12_batch/stoplist.dft
[14] http://boston.lti.cs.cmu.edu/Services/clueweb12_batch/FAQ.html

5. Wikipedia N Sub-state Algorithm

In this section, an outline of the architecture of the WNSSA is described. The WNSSA is designed to 1) select and weight candidate enhancement terms relevant to the user's original search query and, 2) prevent query drift through the use of a dynamically generated stem query which is persisted between internal states. To achieve this, the WNSSA contains two main loops, the *States* loop and the *Sub-states* loop. For each given A state, B number of internal sub-states may run. Algorithm 1 provides a pseudocode example of these loops. At the beginning of this process, a reference cache is first created, described in Section 5.1. When $currentState >= A$, the final enhancement terms are generated.

```
generateCache();
while currentState < A do
    while substateCounter less than B do
        substateCounter = substateCounter + 1;
    end
    currentState = currentState + 1;
end
outputEnhancedQuery();
```

Algorithm 1: WNSSA State and Sub-state Loops

Figure 1 outlines the query and term transition between 5 individual states, each of which have 2 sub-states. The initial query for enhancement in the form of a *root query* is passed into the first State. During this process, additional terms are generated for the query. The top terms from the current state are then passed across to the next state in line. A key element of this process is to persist the original query across to minimise query drift for generated enhancement terms.

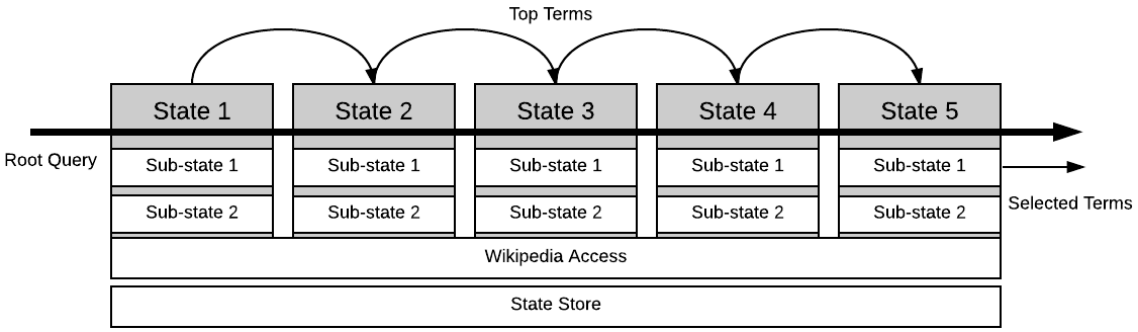

Figure 1: WNSSA State Overview Outlining Five States

The WNSSA is broken into two core steps, *Cache Generation* process described in Section 5.1 and the *Begin Process* described in Section 5.2.

5.1. Inital Cache Generation

Before the algorithm begins, an initial cache must be generated to store information based on the user's original query Q. This cache provides a base of knowledge which can be looked-up as a reference during all sub-states. When generating a cache, query drift is minimised by retaining the user's original search intent. Given a user submitted query Q, Algorithm 2 describes each of the steps which are performed during the cache generation process. At the beginning of this process, simple

processing is performed to remove basic stop words and common Wikipedia function terms from the query Q. By doing this, function words that commonly appear in Wikipedia will be removed from the generation process, preventing undue weighting being attached to these terms.

The *Cache Generation Process* begins by taking the query Q and checking the length. If the length of the query is equal to 1, 2 or 3 tokens long, a connection is made to the Wikipedia article page corresponding to the sequence of terms, including an underscore between each of the terms. The 3 token length was chosen as the threshold as during this research, as well as during analysis of both TREC-9 & TREC 2014 Web Topics, 20% of topics <= 3 tokens had a direct match to a Wikipedia article where, queries > 3 tokens has a 1% direct match. The best performing is single token queries with a 100% match, however single term queries can be ambiguous and context of the search terms would be lost. If no match between the user query and available articles are found, the *failedToFind* flag is set. If the length of the query is greater than 3 tokens or if the *failedToFind* flag is set, then the query is broken down into N-grams of length 2. For each of the N-grams created, the article content gathering process is continued. If after this process, no connections to Wikipedia articles can be made, a flag titled *startSingleTok* is set to 1. This flag then tells the algorithm to break the original query Q into individual tokens, and make a connection to the articles in Wikipedia based on each single token in the query Q.

The *createFilterCache* is responsible for making the connection to the Wikipedia article identified. When a connection is made, the Wikipedia navigation boxes and reference sections are removed from the Article. The main content of the identified article is contained in the *mw-body-content* HTML div tag, which is then selected. Each of the terms in this section are then tokenized into individual single word tokens. If the token is not contained in the Python NLTK English stop word list[15], it is added into the local *filterCache* for future reference. The term frequencies for each token in the article are calculated and stored. To further the understanding of the data that has been returned during the search process, the *GetRootQueryBacklinks* function is called. For each of the tokens in the user submitted query Q, each of the backlinks for each term are retrieved from the Wikipedia Backlink API and stored for use by the data utilisation modules (described in Section 5.3). After this initial cache has been generated, a call is made to the *BeginProcess* function.

5.2. BeginProcess Function

After the initial cache generation process, the *BeginProcess* function is then called. This function is the core of the WNSSA which is responsible for running the defined number of states and sub-states. Algorithm 3 provides a pseudocode outline of this function. While inside one of the defined states in the system, B number of sub-states are run. For each of the individual states which are run, the variable *substateCounter* is used to track the number of sub-states which have run, up to B. The higher the variable B is, the more terms will be added into the sub-state allowing for a wider collection of terms to be utilised during the enhancement term selection process. This can often cause an issue if set too high, as terms of a broader domain may be included.

For each of these sub-states, a term from the previous state is utilised. The *downloadSearchResultsForTerm* begins by taking the stem query (original query, Q plus term under analysis T) and querying the Wikipedia Search API to return records which contain the user's original query with the current term under analysis. The result collection is then tokenized and a term frequency is calculated. If the term is not a stop word in the NLTK English stop word list and is not a Wikipedia function word, the remaining highest weighted term is selected. The *getWeightForTerm* is then passed the current term. If the result is greater than 1, the term was in the original cache created

[15] http://www.nltk.org/

```
removeStopwords(Q);
removeCustomStopwords(Q);
fullQueryLength(len(Q));
failedToFind = 0;
startSingleTok = 0;
if len(Q) less or equal to 3 then
    failedToFind = createFilterCache(Q);
    getRootQueryBacklinks(Q);
end
if len(Q) greater than 3 OR failedtoFind == 1 then
    der = ngram(Q, 2);
    for d in der do
        startSingleTok = createFilterCache(d);
        getRootQueryBacklinks(d);
    end
end
if startSingleTok == 1 then
    der = ngram(Q, 1);
    for d in der do
        createFilterCache(d);
        getRootQueryBacklinks(d);
    end
end
```

Algorithm 2: Cache Generation Process

and is suitable to enter into the enhancement process. This approach prevents query drift from occurring.

```
while currentState < A do
    while substateCounter less than B do
        downloadSearchResultsForTerm(term);
        compareWeight = getWeightForTerm();
        if compareWeight greater than 0 then
            mod1_getTFForPage(term);
            mod2_calculateRecall(term);
            mod3_IntersectionOfQueryToTerms(term) ;
            mod4_backlinkIntersection(term) ;
            mod5_termIntersection(term) ;
            mod6_resultpage_link_sim(term) ;
            substateCounter = substateCounter + 1;
        end
    end
    currentState = currentState +1 ;
    qr = calculateNextQuery();
    if currentState < maxStates+1 then
        beginProcess(qr)
    end
end
```

Algorithm 3: *BeginProcess* function

As the selected term is deemed relevant, each of the six *data utilisation modules*, described in Section 5.3, are called for that term. These modules are responsible for a complete IR process,

gathering information from Wikipedia, and weighting the data collected in accordance to the method defined by each application specific module. These modules are designed so that each utilise a different aspect of the available Wikipedia data set. At the end of the each state, after all the defined B number of sub-states have run and generated their own weightings for each term found, the *calculateNextQuery* function is called. This is an important element of the WNSSA algorithm, which takes in the original user query Q and the collection of the highest weighted terms from the current state. Both the original query and these terms together are passed as a complete query to the next state in the form of a *stem query*. This function is further expanded in Section 5.4.

5.3. Data Utilisation Modules

The WNSSA consists of six different data utilisation modules to gather and weight information gleaned from Wikipedia and the Wikipedia API. These modules include: *Module 1 - Term Frequency* which is focused on generating term frequency statistics for a given term T with the original query Q. A call is made to the Wikipedia Search API and the returned page is parsed and for each term, a check is made to ensure that the term has not already been used and is not a stop word or function word. For each term found the term frequency is calculated and a new record is then stored into the local database. *Module 2 - Wikipedia Recall* was developed to utilise Recall statistics available from Wikipedia. Given a term T and the original user query Q together a query is performed on the Wikipedia database to gather the number of results for term T. This score provides a useful indication as the number of articles where the term has been seen in context with the user's original submitted query. For all terms which are passed to *Module 3 - Backlink Analysis*, a backlink analysis is performed using the Wikipedia backlink API. Given a term T, an exhaustive list of all articles in Wikipedia which are pointing to the article that represents term T in Wikipedia. All links which are gathered are stored in a local database for use by Module 4. Given the collection of backlinks which were generated during the initial cache generation process, a second collection of backlinks for each of the terms are then generated using *Module 4 - Backlink Intersection*. A final score is then created by gathering a list of all backlinks for the user query Q and the term currently under analysis T. The score is calculated based on the intersection of all backlinks shown in Equation 9. In this Equation, the final *score* for a given term T, is calculated by identifying the number of backlinks for query Q that intersect with the backlinks for term T. Results of this module are then stored in the local database.

$$score_{QT} = (backlinks(Q) \cap backlinks(T))) \tag{9}$$

To gauge how important a single term can cause a negative impact on a set of search results, *Module 5 - Intersection of Result Terms* was developed. Two sets of queries are performed, the first collects a set of results based upon the user's original query Q and the second is focused upon the original query with the addition of the term currently being processed. Original query Q and the appended term T are shown in Equation 10.

$$score_{QT} = wikiResults(terms(Q)) \cap wikiResults(terms(Q + T)) \tag{10}$$

The scoring weight is given by the intersection of both sets. The higher the value returned indicates that many of the terms found from the user query Q are similar to those of the query Q with the appended term T. A max number of search results is set to 100 to prevent additional unwanted skew when all search results are not relevant. To assess the similarity of the links in the results which have been returned, *Module 6 - Link Similarity* collects the Wikipedia Article links for the user query Q and the results for query Q and the appended term T. Similar to Module 5,

an intersection of links is then performed giving a final score. This is shown in Equation 11 where Q is the original query and $Q + T$ is the original query plus the term currently under consideration.

$$score_{QT} = wikiResults(links(Q)) \cap wikiResults(links(Q + T)) \tag{11}$$

After each of the six modules have run for the current sub-state and all sub-states have run for the current state, the *stem query* for the next stem query must be generated. This stem query consists of the original query Q in addition to the top terms identified in the current state. The process of identifying the top terms from the current state is done by assessing the weights of all terms generated by each individual module during the current state. As this selection process is the most important step in the WNSSA, it is described in detail in Section 5.4.

5.4. Selecting Stem Query Terms

During the transition between states, a stem query is passed. The stem query generation process begins by select the top terms from each of the data utilisation modules for a given state. Algorithm 4 outlines the process of first looping through each of the modules in the *mod_collection*. For each individual module, the *getStateTerms* function is called, the following parameters are passed: a reference to the current *module* being processed; the *currentState* reference; a *limit* as to how many terms to return; and a final *order* parameter, which in this case, is set to DESC. In this algorithm, the *limit* was set to 2, as if each module returned only a single term, the the probability error is increased if erroneous data is included as a top term for a single module, rendering the weighting for that module useless. If $limit > 2$, too many terms are added into the term selection process.

The *globalTermList* is used to store how many times each of the terms was found for each of the modules. If the term was present in all of the modules, the highest weight of 5 will be given to that term. The *storeTermCollection()* function is then used to store the overall term collection gathered and their associated weights.

```
globalTermList = list();
for mod in mod_collection do
    mod_termCollection = getStateTerms(mod, currentState, limit=2, order=DESC);
    for term in mod_termCollection do
        if term in globalTermList then
            globalTermList[term] = globalTermList[term] + 1;
        end
    end
end
storeTermCollection(globalTermList);
```

Algorithm 4: Creating Window Of Terms

After all of the terms have been processed, Algorithm 5 outlines the process of querying the database to gather a collection of results generated in the previous steps. These results are sorted in descending order by weight. For each term, if they have not been used before during the enhancement process, the term is selected and appended to the new query. The counter variable *num_terms_for_new_query* is used to provide a limit on the number of terms which can be appended to a query for each iteration of the loop. The value for this variable is set to 1 for this research to prevent the drift of the stem query between states.

Terms that are selected to be used as an element of the stem query are appended to the *alreadyUsed* collection. This is used as a reference for future steps to ensure the same terms are not used again in a stem query. At the end of this process, the original user query Q and the new term string are returned together to become the stem query for the next state. To reinforce the use of

```
newTerms = '';
for term in globalTopTerms do
    if term in FilterCache then
        if term not in alreadyUsed then
            if new_term_count < num_terms_for_new_query then
                alreadyUsed.append(term);
                alreadyUsed.append(term + 's');
                newTerms += ' ' + term;
                num_terms_count = num_terms_count + 1;
                if num_terms_count == num_terms_for_new_query then
                    break;
                end
            end
        end
    end
end
return(fullQuery + ' ' + newTerms);
```

Algorithm 5: Building Next Query

1 as the value for *num_terms_for_new_query*, Figure 2 provides an outline of the Average Precision @100 across 50 TREC 2014 Web Topics used as stem queries with enhancement terms generated by the WNSSA using 10 states and 5 sub-states. At the beginning of this process, the original query is the stem query along with additional enhancement terms in increments of 1. In this figure we can see that the larger the *num_terms_for_new_query*, the worse the precision of the result collection from the Wikipedia Search API becomes. In this test, results were considered relevant if each of the terms provided could be seen in the result record. For this reason, and to ensure the collection of records utilised by the WNSSA are relevant and will not run out of source data during processing, the *num_terms_for_new_query* was limited to 1.

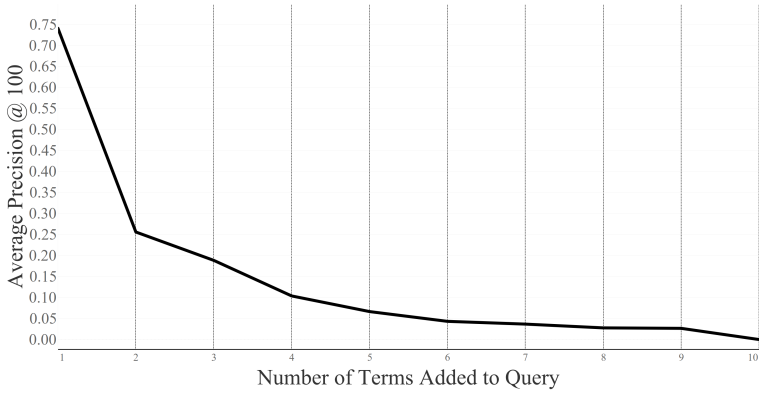

Figure 2: Average Precision @100 Decrease as num_terms_for_new_query is Increased

After each of the A states has been run, and each of the top terms have been generated, a call is made to the *outputEnhancedQuery()* function to generate the final enhanced query. In this function, a query is made to the database to collect a list of the overall top terms across all states by calling the *getOverallTopTerms()* function and passing the *states* variable set to *ALL*. A limit of 10 is passed here to only return 10 enhancement terms. The top terms from the process and the original user query Q are merged and returned to the user. This process is shown in Algorithm 6.

enhancementTerms = getOverallTopTerms(states=All, limit=10);
finalQ = Q + ' ' + enhancementTerms;
return finalQ;

Algorithm 6: outputEnhancedQuery Function

6. Algorithm Comparison Results

For each algorithm including the WNSSA, the Mean Average Precision (MAP) was calculated using the Average Precision (AP) scores for each of the 50 TREC-9 Web Topics on the ClueWeb12 batch query service. Table 1 provides an outline of these results. The *Diff* describes the difference from the MAP score achieved by the WNSSA, the WNSSA Improvement outlines this in percent. The *p-Value* outlines the results of significance testing which was performed using paired T-tests with two tails and an α of 0.05. The over all standard deviation for each algorithm was the calculated on the AP, shown as STD. To provide an understanding of each algorithm for short and long queries the STD, and MAP was calculated for 1 term topics (short), 2 term topics (short) and greater than 2 topics (long).

Table 1: Results from Tested Algorithms

	MAP @10	Diff	WNSSA Improvement	p-Value	STD	1 Term MAP	1 Term STD	2 Term MAP	2 Term STD	$\neg^2$ Term MAP	$\neg^2$ Term STD
WNSSA	0.800				0.350	0.910	0.186	0.897	0.290	0.736	0.423
Algo 1	0.634	-0.165	26%	0.0058	0.415	0.739	0.340	0.536	0.463	0.617	0.438
Algo 2	0.364	-0.436	120%	9.31E-10	0.397	0.559	0.444	0.323	0.337	0.282	0.378
Algo 3	0.694	-0.105	15%	0.060	0.405	0.769	0.365	0.914	0.192	0.549	0.450
Algo 4	0.460	-0.339	74%	1.025E-05	0.467	0.597	0.449	0.464	0.460	0.401	0.479
Algo 5	0.480	-0.319	66%	6.082E-05	0.436	0.595	0.455	0.243	0.413	0.535	0.410
Average		-0.273	60%								

Algorithm 1 by ALMasri et al. (2013) scored 0.634 and had issue that arose from a number of different function words being added into titles. And as no validation is performed, articles are often added without consideration for their relevance. Algorithm 2 described by Boston et al. (2014), scored the lowest at 0.364. Although the terms generated were relevant to the domain and more precision was added into the process of selecting the importance of terms, no real understanding is gained about the query that has been entered by the user. A high dependency was also placed on the initial retrieval. The coverage of the terms can be seen as very broad as the number of documents that are included may cover many different domains. Algorithm 3 described by Xu et al. (2009) provided the best overall performance scoring 0.694, second to the WNSSA. The overall success is likely due to its simplicity. Rather than utilising a traditional TF-IDF approach which uses a document subset during the calculations, the entire Wikipedia collection is used. A useful element of this approach is the focus placed upon using Entity Pages (single individual Articles). This focus prevents completely irrelevant terms being utilised. Algorithm 4 described by Bruce et al. (2012) was the second worst performing with 0.460. The intent of the query has no impact on the

enhancement process. Many of the terms added may have a high score and appear to be highly relevant to the domain but not to the user's query. If the user was focused on gathering terms relevant to a domain, this would be a useful approach. Many terms that have been generated are very useful to a domain, however when multiple terms are added into the query, the diversity of the results become overwhelming, impacting the overall term ranking. Algorithm 5 by Zhao et al. (2014) scored 0.480 and was focused upon the heavy utilisation of inter-wiki links. A heavy dependence is placed on these links. If irrelevant articles are found but have a high number of similar outbound links then these terms will be given a high weighting which can impact the results. Unlike other algorithms, there is a higher distribution of good and bad results for each of the queries.

Table 2 provides an overview of the generated enhancement terms for each of the tested Algorithms for TREC-9 Topics 4, 17, 31 and 32. In this table we can see that for Topic 4 *parkinsons disease*, the WNSSA focused on each of the different aspects of the disease such as related foundations, treatment and drugs. Algorithm 2, had only a single related term, neurology. Algorithm 3 included the initials pd, and different elements related to the disease. Algorithm 4, had similar results to the WNSSA. Algorithm 5 however included terms such as tuberculosis and pathology, which are not directly related. Topic 17: *dachshund dachshunds "wiener dog"*, the WNSSA included elements related to the dogs, such as the sport of dachshund racing. Algorithm 2 produced poor results not related to the topic. Algorithm 3, focused on different breeds which are available. Algorithm 3 again focused upon different breeds, and algorithm 5 added in two terms which were irrelevant, "comedy" and "deer". Topic 31: *What did Babe Ruth do in the 1920's*, has very narrow room for error as it is specifically looking for one topic, Baseball. In the results, we can see that baseball was shown for the WNSSA. Algorithm 2, included terms such as *Curse of the Bambino*, which is related to Babe Ruth, and the name of baseball teams. However, no explicit reference to baseball was included. Algorithm 2, performed poorly overall. Algorithm 3 focused again on the term Baseball and outlined other teams and notable names in Baseball. Algorithm 4 did not produce any useful enhancements. Algorithm 5 again outlined baseball, pitcher and outlined 1920's related topics such as prohibition. Topic 32, *where can i find the growth rate for the pine tree?* showed successful enhancements for the WNSSA with terms such pines, roots, rapid, high. Algorithm 1 produced terms such as Christmas tree, which can be deemed irrelevant, hurting precision. Algorithm 2 produced, terms such as nigra and petals which can impact the precision. Algorithm 3, produced terms which are relevant to trees such as pinus and cones, Algorithm 5 focused on the genus of trees, which although relevant can hurt the intent of the query.

In the TREC-9 Web topic collections, examples of queries which performed poorly across all implementations include Topic 2: *do beavers live in salt water* and Topic 24: *how email benefits business*, this can be attributed do the question based nature of this query, a number of different concepts are all represented inside the same query. This is in contrast to topics such as Topic 40 *motorcycle safety helmets* & Topic 26 *Jennifer Aniston*, which represent one single core topics.

From the testing of these five existing ASQE algorithms that utilise Wikipedia, the following issues with Wikipedia powered ASQE across each algorithm can be seen: 1) The utilisation of documents as sources of knowledge without proper understanding of the domain of the documents that are being added. 2) No validation of candidate terms as being relevant or irrelevant is made. 3) Content in documents which are not relative to the article, e.g., advertisements, long additional text descriptions are often utilised. 4) Many of the algorithms allow function words or pages which are relevant to Wikipedia to appear in enhancement terms. 5) An over dependence on the weights that have been assigned to the terms without additional processing being performed. Although some terms may be very high quality, a post processing stage would greatly improve the overall success of the enhancements. 6) Many search queries are entered by users are in the form of a question. Many of the terms that are added in a question format have an impact on the results which are returned,

Table 2: Sample Generated Enhancement Terms for Each Tested Algorithm using TREC-9 Topics

	Topic 4: parkinson's disease
WNSSA	parkinsons disease foundation treat target research treatment shown drugs people symptoms disorders
Algo 1	Lewy body disease Parkinson disease 2 American Parkinson Disease Association
Algo 2	home high group including university samii neurology increase list association
Algo 3	pd symptoms levodopa pmid doi dopamine motor disorder lewy brain
Algo 4	dopamine parkinsonism cases therapy sleep studies medication disord although system
Algo 5	neurology parkinsonism psychiatric idiopathic symptom pathology infectious pain tuberculosis pathogenic
	Topic 17: dachshund dachshunds "wiener dog"
WNSSA	dachshund wiener dog pet racing dogs named american america lives hot lm dachshunds
Algo 1	Fatal dog attacks in the United States Capitalist pig-dog
Algo 2	result recognized making entering ramirez chase companion essays heed quirk
Algo 3	dapple kennel breed wire-haired miniature akc teckel piebald standard anglo-fran
Algo 4	wire-haired anglo-franxc long-haired smooth-haired california short-haired full-size long-bodied double-dapple merriam-webster
Algo 5	kennel breed comedy deer scent
	Topic 31: what did babe ruth do in the 1920's?
WNSSA	babe ruth 1920s yankees baseball home season park yankee became late gehrig teams
Algo 1	Harmonica Incident Curse of the Bambino Charlie Gehringer The Yankees
Algo 2	yugoslav patrol soap bob miguel arkansas pioneer sabina pop prime
Algo 3	yankees creamer montville baseball sox wagenheim home runs reisler gehrig
Algo 4	economic publishes minister prime republic europe political fascist william cricketer
Algo 5	pitcher manhattan outelder boston baseball surage millennium prohibition decade ratication
	Topic 32: where can i find growth rates for the pine tree?
WNSSA	growth rates pine tree high rapid caused old pines species vegetation short trunk roots
Algo 1	Felled tree Taiga Silviculture Christmas tree Maine Everglades National Park
Algo 2	relatively protein spirally shell control internal herbaceous physiology nigra petals
Algo 3	pinus cones wood needles species seeds r sp pinyon acacia
Algo 4	population theory directly property landau cagr measure produced notation economy
Algo 5	pinus subgenus fir foliage genus

however, many of the algorithms ignore these terms. 7) Although in many cases, a reference to the original submitted query is used as a stem for the identification of relevant content, no call back is made in future steps to identify if the terms generated are links back to the original user's query. 8) Query drift appears in all algorithms, especially in longer queries, as query terms are often expanded independently of the rest of the terms in the search query. 9) Shorter search queries often lack context and are not treated with care allowing irrelevant expansion terms to be included.

6.1. WNSSA Time Efficiency

To provide an understand of the time efficiency of the WNSSA algorithm, Table 3 outlines the time in seconds on a Amazon Web Services (AWS) T2.medium Instance[16] for each run with variable States and Sub-states. This AWS instance contains 2 vCPU and 4 GiB of Memory. Times were calculated for each individual run using the Python time package, calculating the total run of each script from start to completion. In these result, we can see a linear time increase as the number of states and sub-states increase for each run. This can be attributed to the constant revisiting to the Wikipedia API, which is computationally expensive. As this study focuses on the implementation of the WNSSA algorithm, no further analysis into the time-efficiency optimisation of this algorithm has been included in this study.

Table 3: WNSSA Time Efficiency in Seconds

	Sub-state 1	Sub-state 2	Sub-state 3	Sub-state 4	Sub-state 5	Sub-state 6	Sub-state 7	Sub-state 8	Sub-state 9	Sub-state 10
State 1	86	96	108	119	133	146	161	178	193	212
State 2	108	130	148	181	212	235	262	283	316	352
State 3	123	159	195	244	275	327	368	397	442	494
State 4	137	202	255	287	327	395	453	506	571	635
State 5	148	226	299	357	414	467	539	609	693	1,007
State 6	215	343	413	417	447	553	625	728	821	931
State 7	178	258	377	441	538	625	726	825	938	1,048
State 8	190	305	415	488	585	684	776	892	1,010	1,165
State 9	211	306	476	586	671	775	915	1,052	1,232	1,350
State 10	226	356	480	602	707	858	1,026	1,160	1,341	1,498

6.2. Optimal States and Sub-States Variables

To provide an insight into the performance of the WNSSA with different State and Sub-state variations, 30 individual tests were performed using the TREC 2014 Web Topics on the ClueWeb12 full data set. Table 4 outlines the Test ID along with the associated state and sub-state parameters. 15,000 individual manual relevance assessments were performed during this testing, of which each result was marked 1 for relevant and 0 for not relevant for each of the 50 queries of the top 10 returned results. For each test, the Mean Average Precision (MAP) scores for each test were then calculated.

Table 5 provides an overview of the MAP score achieved by each test. Results are shown in descending order of MAP scores by performance. In these results we can see that Tests T1, T4,

[16]https://aws.amazon.com/ec2/instance-types/

Table 4: WNSSA State and Sub-state Test Parameters Variations

Test ID	State	Sub-state	Test ID	State	Sub-state	Test ID	State	Sub-state	Test ID	State	Sub-state	Test ID	State	Sub-state	Test ID	State	Sub-state
T1	1	1	**T6**	10	2	**T11**	5	4	**T16**	1	6	**T21**	10	7	**T26**	5	9
T2	5	1	**T7**	1	3	**T12**	10	4	**T17**	5	6	**T22**	1	8	**T27**	10	9
T3	10	1	**T8**	5	3	**T13**	1	5	**T18**	10	6	**T23**	5	8	**T28**	1	10
T4	1	2	**T9**	10	3	**T14**	5	5	**T19**	1	7	**T24**	10	8	**T29**	5	10
T5	5	2	**T10**	1	4	**T15**	10	5	**T20**	5	7	**T25**	1	9	**T30**	10	10

T7, T24 and T11 achieved the highest MAP scores. For the top 10 performing results, the average state value is 5.3 with a standard deviation of 3.68 and the average number of sub-states is 3.5 with a standard deviation of 2.27. Across all 30 tests, the average MAP score is 0.732, with a Standard Deviation of 0.055.

Table 5: WNSSA Variable State and Sub-state MAP Results

Test ID	Result	Test ID	Result	Test ID	Result	Test ID	Result	Test ID	Result	Test ID	Result
T1	0.912	T6	0.763	T14	0.745	T16	0.726	T13	0.714	T26	0.684
T4	0.827	T20	0.757	T23	0.739	T22	0.724	T27	0.706	T19	0.683
T7	0.810	T5	0.755	T15	0.736	T2	0.722	T29	0.705	T30	0.674
T24	0.769	T8	0.752	T21	0.735	T18	0.721	T12	0.691	T25	0.653
T11	0.765	T9	0.747	T17	0.735	T10	0.71746	T3	0.689	T28	0.609

7. Conclusion

This paper outlined the algorithm and testing of the proposed ASQE algorithm, the Wikipedia N Sub-state Algorithm (WNSSA). The proposed algorithm was tested against five existing ASQE algorithms that utilise different aspects of the Wikipedia data set as the sole data source for enhancement. The WNSSA was designed to provide a unique state based approach to ASQE which allows re-occurring terms across a number of states to be identified as the most suitable enhancement terms. During the execution of the WNSSA, a defined number of states are run which in turn run a set amount of internal sub-states. Throughout the process of running each sub-state, six data utilisation modules are tasked with collecting data from Wikipedia and applying term weighting metrics to each of the terms found.

Between the transition of different states inside the WNSSA, a stem query is created which takes the user's original query along with the top selected term from the current state to be passed onto the next state. The top term for a given state is selected through the unique approach of a custom *Term Window* which gathers all terms generated by each sub-state module and selects the term which appears most frequently across all modules for that given state. This approach allowed only the highest quality term to persist to the next state. The WNSSA utilises a custom cache which is generated from the originally submitted query by n-gramming the query and making a discovery of Wikipedia articles where the sub-query is found in the title. This approach allowed for a narrow collection of content to be referenced throughout the life-cycle of the WNSSA referencing the user's original intent in content form. After the WNSSA was developed, 50 of the TREC-9 Web Topics

were enhanced using the WNSSA and run on the ClueWeb12 data set. The WNSSA had an average MAP improvement of 0.273 over existing algorithms, scoring an overall MAP of 0.800.

During this process, the WNSSA showed to outperform all of the five existing algorithms tested during the benchmark analysis. The WNSSA has shown to perform well on both short and long queries where the query represents one main concept. Unlike the tested benchmark algorithms, the WNSSA had the highest number of complete successful identified results losing only when a rare complete failure of expansion was performed. The cause of this can be attributed to the lack of background and contextual information available to ASQE techniques alike. Results of testing the WNSSA have shown that:

- By narrowing down the data available to the ASQE algorithm less skew can appear in the enhancement term collection.
- Keeping a constant point of contact with the user's original query as a stem can be beneficial to the enhancement process as the intent is never lost.
- Blind use of any relevance weighting schemes should never be performed and alternative validation should always be included in the process to ensure no rogue terms enter the enhancement process.
- A state based approach to enhancement provides a number of opportunities for enhancement terms that have frequently been appearing to get into the final collection of enhancement terms.
- A term window based approach to term selection avoids the need for raw multiplication of generated relevance weighting schemes.
- Taking simple but effective methods of weighting terms can be further utilised by deriving intersections and statistics from a wider data collection, in the form of Wikipedia.
- Generating a cache of content relevant to the user's original query can help persist the intent of the user's query by utilising the cache as a filter for future expansion terms.

8. Future Work

As the WNSSA does not utilise user context, issues can arise when understanding areas of interest for the user. Using additional contextual information such as queries gathered from the user's browser or search session can be invaluable for preventing wrongful expansion when difficulties arise. The current implementation of the WNSSA has not been parallelised, doing so would greatly enhance the overall time efficiency of the algorithm. In addition to this, the use of memory caching of data stored in the local database would also enhance the algorithm performance.

References

Al-Shboul, B., & Myaeng, S.-H. (2014). Wikipedia-based query phrase expansion in patent class search. *Information Retrieval*, *17*, 430–451. URL: http://dx.doi.org/10.1007/s10791-013-9233-4. doi:10.1007/s10791-013-9233-4.

ALMasri, M., Berrut, C., & Chevallet, J.-P. (2013). Wikipedia-based semantic query enrichment. In *Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval* ESAIR '13 (pp. 5–8). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/2513204.2513209. doi:10.1145/2513204.2513209.

Boston, C., Fang, H., Carberry, S., Wu, H., & Liu, X. (2014). Wikimantic: Toward effective disambiguation and expansion of queries. *Data & Knowledge Engineering*, *90*, 22 – 37. URL: http://www.sciencedirect.com/science/article/pii/S0169023X13000761. doi:http://dx.doi.org/10.1016/j.datak.2013.07.004. Special Issue on Natural Language Processing and Information Systems (NLDB 2012).

Bruce, C., Gao, X., Andreae, P., & Jabeen, S. (2012). Query expansion powered by wikipedia hyperlinks. In M. Thielscher, & D. Zhang (Eds.), *AI 2012: Advances in Artificial Intelligence: 25th Australasian Joint Conference, Sydney, Australia, December 4-7, 2012. Proceedings* (pp. 421–432). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: http://dx.doi.org/10.1007/978-3-642-35101-3_36. doi:10.1007/978-3-642-35101-3_36.

Fourney, A., & Dumais, S. T. (2016). Automatic identification and contextual reformulation of implicit system-related queries. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '16 (pp. 761–764). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/2911451.2914701. doi:10.1145/2911451.2914701.

Habibi, M., Mahdabi, P., & Popescu-Belis, A. (2016). Question answering in conversations: Query refinement using contextual and semantic information. *Data & Knowledge Engineering*, *106*, 38 – 51. URL: http://www.sciencedirect.com/science/article/pii/S0169023X16300489. doi:http://dx.doi.org/10.1016/j.datak.2016.06.003.

He, B., & Ounis, I. (2009). Studying query expansion effectiveness. In M. Boughanem, C. Berrut, J. Mothe, & C. Soule-Dupuy (Eds.), *Advances in Information Retrieval: 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings* (pp. 611–619). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: http://dx.doi.org/10.1007/978-3-642-00958-7_57. doi:10.1007/978-3-642-00958-7_57.

Jiang, Y., Zhang, X., Tang, Y., & Nie, R. (2015). Feature-based approaches to semantic similarity assessment of concepts using wikipedia. *Information Processing & Management*, *51*, 215 – 234. URL: http://www.sciencedirect.com/science/article/pii/S0306457315000023. doi:http://dx.doi.org/10.1016/j.ipm.2015.01.001.

Karisani, P., Rahgozar, M., & Oroumchian, F. (2016). A query term re-weighting approach using document similarity. *Information Processing & Management*, *52*, 478 – 489. URL: http://www.sciencedirect.com/science/article/pii/S030645731500120X. doi:http://dx.doi.org/10.1016/j.ipm.2015.09.002.

Monchaux, S., Amadieu, F., Chevalier, A., & Marin, C. (2015). Query strategies during information searching: Effects of prior domain knowledge and complexity of the information problems to be solved. *Information Processing & Management*, *51*, 557 – 569. URL: http://www.sciencedirect.com/science/article/pii/S0306457315000552. doi:http://dx.doi.org/10.1016/j.ipm.2015.05.004.

Ogilvie, P., Voorhees, E., & Callan, J. (2009). On the number of terms used in automatic query expansion. *Information Retrieval*, *12*, 666. URL: http://dx.doi.org/10.1007/s10791-009-9104-1. doi:10.1007/s10791-009-9104-1.

Roy, R. S., Agarwal, S., Ganguly, N., & Choudhury, M. (2016). Syntactic complexity of web search queries through the lenses of language models, networks and users. *Information Processing & Management*, *52*, 923 – 948. URL: http://www.sciencedirect.com/science/article/pii/S0306457316300528. doi:http://dx.doi.org/10.1016/j.ipm.2016.04.002.

Selvaretnam, B., & Belkhatir, M. (2016). A linguistically driven framework for query expansion via grammatical constituent highlighting and role-based concept weighting. *Information Processing & Management*, *52*, 174 – 192. URL: http://www.sciencedirect.com/science/article/pii/S0306457315000473. doi:http://dx.doi.org/10.1016/j.ipm.2015.04.002.

Sharma, P., Tripathi, R., & Tripathi, R. (2015). Finding similar patents through semantic query expansion. *Procedia Computer Science*, *54*, 390 – 395. URL: http://www.sciencedirect.com/science/article/pii/S1877050915013691. doi:http://dx.doi.org/10.1016/j.procs.2015.06.045.

Shtok, A., Kurland, O., Carmel, D., Raiber, F., & Markovits, G. (2012). Predicting query performance by query-drift estimation. *ACM Trans. Inf. Syst.*, *30*, 11:1–11:35. URL: http://doi.acm.org/10.1145/2180868.2180873. doi:10.1145/2180868.2180873.

Vilares, J., Alonso, M. A., Doval, Y., & Vilares, M. (2016). Studying the effect and treatment of misspelled queries in cross-language information retrieval. *Information Processing & Management*, *52*, 646 – 657. URL: http://www.sciencedirect.com/science/article/pii/S0306457315001478. doi:http://dx.doi.org/10.1016/j.ipm.2015.12.010.

Xu, Y., Jones, G. J., & Wang, B. (2009). Query dependent pseudo-relevance feedback based on wikipedia. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '09 (pp. 59–66). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/1571941.1571954. doi:10.1145/1571941.1571954.

Yadav, V., & Kumar, S. (2016). Learning web queries for retrieval of relevant information about an entity in a wikipedia category. In *Proceedings of the 25th International Conference Companion on World Wide Web* WWW '16 Companion (pp. 1013–1014). Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee. URL: http://dx.doi.org/10.1145/2872518.2891114. doi:10.1145/2872518.2891114.

Zhao, D., Liu, P., Qin, L., & Li, Y. (2014). A novel terms semantic query model based on wikipedia. In *2014 11th Web Information System and Application Conference* (pp. 258–261). doi:10.1109/WISA.2014.54.

Zhao, G., Wu, J., Wang, D., & Li, T. (2016). Entity disambiguation to wikipedia using collective ranking. *Information Processing & Management*, *52*, 1247 – 1257. URL: http://www.sciencedirect.com/science/article/pii/S0306457316301893. doi:http://dx.doi.org/10.1016/j.ipm.2016.06.002.

Zhao, L., & Callan, J. (2012). Automatic term mismatch diagnosis for selective query expansion. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '12 (pp. 515–524). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/2348283.2348354. doi:10.1145/2348283.2348354.

Zingla, M. A., Chiraz, L., & Slimani, Y. (2016). Short query expansion for microblog retrieval. *Procedia Computer Science*, *96*, 225 – 234. URL: http://www.sciencedirect.com/science/article/pii/S1877050916319366. doi:http://dx.doi.org/10.1016/j.procs.2016.08.135. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016.