

Markovian Semantic Relatedness

a Novel Approach to Content Discovery

Karl Preisner, Claire Digirolamo, Bria Flowers, Benjamin Breech, and Jon Kosloski

April 25, 2022

Abstract

Query expansion can be a powerful tool for solving the problem of query-document mismatch in information retrieval. We designed a novel technique, which we call Markovian Semantic Relatedness (MSR), that utilizes unsupervised machine learning paired with a Markov chain graph traversal to perform weighted query expansions for content discovery. Applications include performing a ranked, semantic search of information at an arbitrary, sub-document level. MSR is data type agnostic and is designed to work on a variety of structured and unstructured data.

1 Introduction

The problem of effectively answering a user’s queries in information retrieval has been well explored. Query expansion, the process of adding terms to a user’s keyword query to return more relevant documents, is one popular approach [22]. Existing methods for query expansion generally focus on natural language processing (NLP) and rely on ontologies or other syntactic rules to obtain relevant results [1, 8, 13, 17, 19]. In this paper we outline MSR, our data agnostic query expansion approach for content discovery, which is aimed at introducing users to relevant information.

Many existing query expansion methodologies rely only on semantic similarity in order to improve results. As the authors in [18] write, semantic relatedness “is a more general notion of the relatedness of concepts” whereas semantic similarity is a “special case of relatedness that is tied to the likeness (in the shape or form) of the concepts.” A more nuanced explanation of semantic relatedness is provided by the authors of [10], who define the ideas of thematic and taxonomic relatedness. “Taxonomic relations between concepts are based on a comparison of the concepts’ features; concepts that belong to a common taxonomic category share properties and/or functions, and therefore tend to bear physical resemblance. In contrast, thematic relations are formed between concepts performing complementary roles in a common event or theme, which often implies having different (albeit complementary) features and functions.”

A traditional query expansion might rely on taxonomic relatedness and provide synonyms to the user’s original query. Our goal in developing the MSR algorithm is to incorporate thematic and taxonomic relatedness to provide the complete picture of semantic relatedness. Thus the root term “cat” using MSR would return both the words “kitten” and “veterinarian”. In this case “kitten” is taxonomically related to “cat”, while “veterinarian” would be thematically related.

MSR solely relies on statistical co-occurrences of terms within chunks of data, enabling it to be data type agnostic. The key aspect of MSR’s effectiveness is the combination of its term-term model definition, a Markovian graph traversal, an expected hitting time calculation, and a z-score measure of relevance. MSR allows one to infer term relationships from corpora of data without needing to tailor the algorithm specifically for natural language. Furthermore, MSR query expansions excel at discovering semantically related terms, reinforce subtle relationships, and eliminate irrelevant and common terms.

We note the use of random walks in query expansion is not new [2, 3, 19]. However, these other methods either rely on predetermined language graphs such as Wordnet, or use metrics such as PageRank and cosine similarity to measure relevance. Query expansion systems trained on thesauri or general ontologies may be ill suited for specific applications, such as expanding queries for scientific or biomedical terminology, and require specialized strategies [4, 21].

MSR is able to avoid large amounts of data preprocessing; it only requires unlabeled training data that is chunked and tokenized. Additional preprocessing such as performing coreference resolution, stemming, or tagging parts of speech on text data can provide more precise results but is not needed for obtaining decent results. State-of-the-art neural approaches, such as BERT models, can be extremely effective at finding semantic relationships. However, they require large amounts of training data and can require intense computational resources [9, 15, 23]. MSR can create effective models out of small amounts of data, scaling down the computational cost.

While many methods use data chunks to develop contextual embeddings, the chunks tend to be arbitrarily-sized groups of tokens [6, 5, 14, 15]. MSR favors using adaptive chunking. For natural language data, we prefer to segment documents at the paragraph level as it emulates human intuition for how terms are naturally grouped. Additionally, this enables us to more easily capture semantic relationships without performing additional preprocessing, like coreference resolution.

The remainder of this paper explores our approach to MSR’s weighted query expansion and its applications. In Section 2, we give an overview of the MSR algorithm. In Section 3, we discuss our data preprocessing and unsupervised model-building approach. In Section 4, we present our approach to generating the weighted query expansion, which we refer to as a *concept*. In Section 5, we discuss our approach for semantically searching data with multiple concepts and give examples for MSR discovery. We also include examples for several of MSR’s various applications: MSR discovery, MSR discovery across data types, MSR cross-language query expansion, and MSR cross-language semantic search on untranslated, natural language data at an arbitrary, sub-document level. Finally, our future work is outlined in in Section 6.

2 Overview

MSR begins by creating a *model*: a weighted, directed graph that is a function of the concurrence relationships between all unique terms in a corpus of data. The user selects a root term from the model that MSR then uses to perform a Markovian graph traversal of the model, creating a concept. The concept is a weighted list describing how every term in the model is related to the root term, where a higher weight indicates that a term is more relevant to the root term. This is designed to reflect how relevant documents might not explicitly contain the user’s search term. For example, the term “lion” might get a relatively high weight in the concept created for “cat.” If a user was searching for any information about cats, a paragraph discussing lions would meet the criteria of the search even if the term “cat” didn’t appear explicitly. The resulting concept can then be used to perform a ranked, semantic search on any repository of data, not just the training data used to construct the model.

MSR concepts excel at finding subtle relationships in the data while simultaneously filtering out irrelevant terms. Rather than a traditional Boolean search, which weights all search terms equally, MSR is able to semantically search data and sort results relative to the weighted concept terms, enabling users to better discover relevant pieces of information. Figure 1 illustrates the components of the MSR algorithm, as well as alternative approaches we have previously considered.

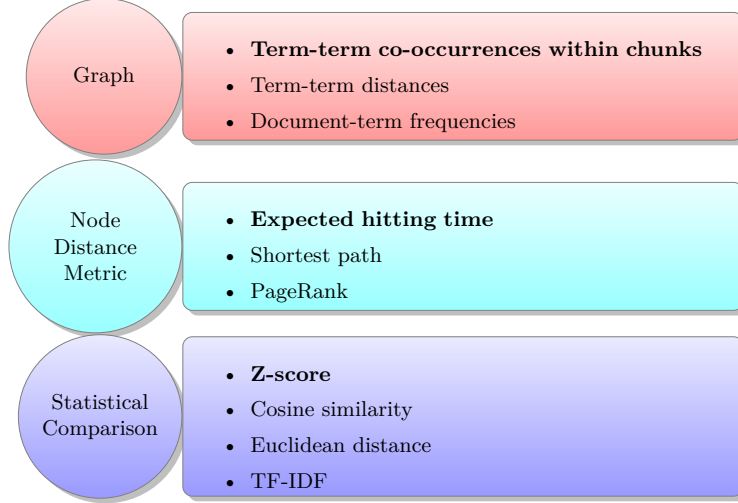


Figure 1: Diagram of MSR components and alternative approaches.

3 Building the Model

To create the graph that will become the MSR model, we begin with a repository of processed, tokenized training data. The data are subdivided into smaller units that we refer to as chunks. For example, in text documents, we create chunks out of paragraphs. Because MSR is an unsupervised method, we are able to use unstructured and unlabeled data. MSR will use the model graph to find the relationships between terms in the data without user input. Model creation is divided into two steps. We begin by creating the concurrence matrix, $\mathbf{C}$, then the state transition probability matrix, $\mathbf{S}$. Matrix $\mathbf{S}$ is the MSR model graph. In practice, $\mathbf{S}$ may need to be approximated for a feasible computational implementation. We call this approximated matrix $\mathbf{T}$; its definition is an active area of research.

3.1 Data Preprocessing

Before we create a model, we must perform initial preprocessing on the training data. Data preprocessing consists of two main steps: separating the data into chunks and finding the unique set of terms within each chunk. For unstructured, natural language data, we create chunks out of entire paragraphs rather than individual sentences. We find this approach produces better results with MSR concept creation as it more closely mimics human intuition: ideas tend to be separated into paragraphs and terms that co-occur in a given paragraph are likely to be meaningful to one another. Separating data into chunks at the paragraph level instead of the sentence level also removes the need for coreference resolution. We then find the unique set of terms in each chunk of training data. For unstructured, natural language data, we tokenize the individual words. We suggest adding a layer of named entity recognition (NER) to aid the separation of unique terms. This process does not track the distances between words or characters in a chunk, nor the order that terms appear.

Consider the example of training data in Figure 2. For this example, we consider each line to be a single chunk. We tokenize the chunks to obtain the set of unique terms in each chunk. These steps are illustrated in Figure 3. The use of NER results in “Bill G” being recognized as a full name, so “Bill” and “Bill G” appear as separate, unique terms in *chunk 0*. We represent this set of tokenized chunks graphically in Figure 4 by connecting unique terms within chunks.

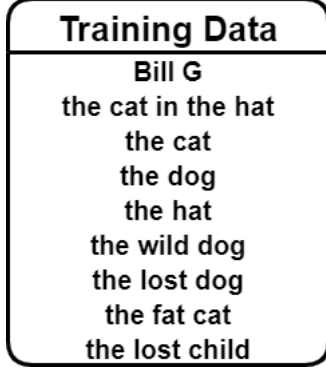


Figure 2: Corpus of training data for the small example.

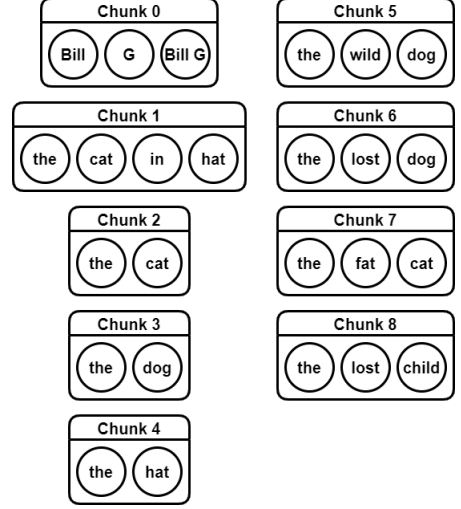


Figure 3: Tokenized chunks for the small example.

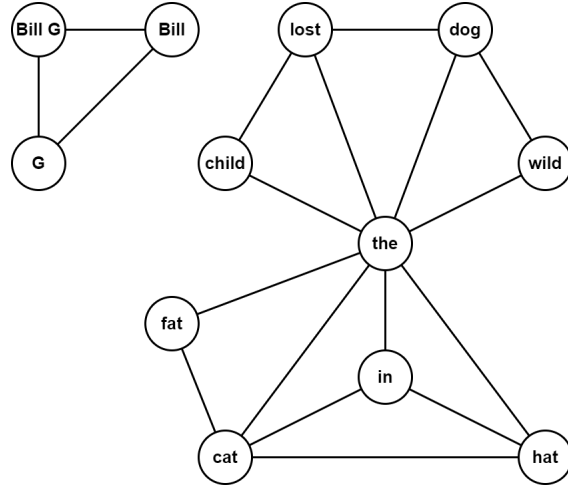


Figure 4: Graphical representation of the small example tokenized chunks in Figure 3.

3.2 Building the Concurrence Matrix

Let Ω be the corpus of training data and let N be the number of unique terms found within the pre-processed training data. We first construct the term-term concurrence matrix, $\mathbf{C}$, which is an $N \times N$ symmetric matrix. The entry $C_{i,j}$ represents the number of chunks where term $i \in \Omega$ co-occurs with term $j \in \Omega$ and $C_{i,i}$ is the number of chunks where the term $i \in \Omega$ appears. The $\mathbf{C}$ matrix is a representation for an undirected graph for term co-occurrences within chunks. The naïve algorithm for constructing $\mathbf{C}$ is as follows:

```

Data: DataChunksList
Result: Concurrence matrix,  $\mathbf{C}$ 
for  $DataChunk$  in  $DataChunksList$  do
  for  $term_i$  in  $DataChunk$  do
     $C[term_i, term_i] \leftarrow C[term_i, term_i] + 1$ 
    for  $term_j$  in  $DataChunk$  do
      if  $term_i \neq term_j$  then
         $C[term_i, term_j] \leftarrow C[term_i, term_j] + 1$ 
      else
        continue
      end
    end
  end
end

```

The above algorithm iterates over every individual chunk in the processed training data and increments the edge weight in $\mathbf{C}$ by 1 for every pair of unique terms within the chunk. In practice, we take advantage of the symmetry of $\mathbf{C}$ and initially only build the upper triangle of the matrix, which allows us to save computational time and space.

Consider the previous small example of tokenized chunks in Figure 3. The concurrence matrix, $\mathbf{C}$, for this set of data is

$$\begin{bmatrix}
 & \text{Bill} & \text{Bill G} & \text{G} & \text{cat} & \text{child} & \text{dog} & \text{fat} & \text{hat} & \text{in} & \text{lost} & \text{the} & \text{wild} \\
 \text{Bill} & \mathbf{1} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 \text{Bill G} & 1 & \mathbf{1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 \text{G} & 1 & 1 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 \text{cat} & 0 & 0 & 0 & \mathbf{3} & 0 & 0 & 1 & 1 & 1 & 0 & 3 & 0 \\
 \text{child} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
 \text{dog} & 0 & 0 & 0 & 0 & 0 & \mathbf{3} & 0 & 0 & 0 & 1 & 3 & 1 \\
 \text{fat} & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 1 & 0 \\
 \text{hat} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{2} & 1 & 0 & 2 & 0 \\
 \text{in} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 1 & 0 \\
 \text{lost} & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \mathbf{2} & 2 & 0 \\
 \text{the} & 0 & 0 & 0 & 3 & 1 & 3 & 1 & 2 & 1 & 2 & \mathbf{8} & 1 \\
 \text{wild} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \mathbf{1}
 \end{bmatrix}. \quad (1)$$

We can see that (1) is a sparse, 12×12 symmetric matrix where $N = 12$ is the number of unique terms in the training data. The entries of (1) can be thought of as the weighted edges of an undirected graph representing the relationship between terms in the data. We present a graph of this matrix in Figure 5; we note the graph has two connected components, which will be important when we perform the graph traversal in Section 4.

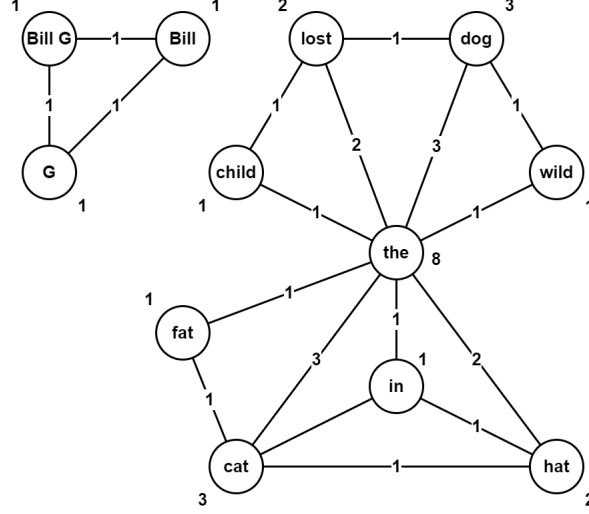


Figure 5: Graphical representation of the $\mathbf{C}$ matrix in (1). We note that the numbers adjacent the graph nodes are the weights of the self-edges. The self-loops are omitted from the diagram for simplicity.

3.3 The Standardized Matrix

From the concurrence matrix, $\mathbf{C}$, we construct the standardized matrix, $\mathbf{M}$. Entries in $\mathbf{M}$ are defined as

$$\begin{cases} M_{i,j} = \frac{C_{i,j}}{C_{j,j}}, & \text{for } i \neq j \\ M_{i,j} = 0, & \text{for } i = j \end{cases} \quad (2)$$

where $M_{i,j}$ represents the approximate probability that when term j appears in a chunk, term i also appears in that chunk. We remove the possibility of self-transitions by setting entries of $\mathbf{M}$ where $i = j$ to zero; this excludes the probabilities of terms occurring with themselves. We now have a directed graph where each node has one in-edge and one out-edge for every node that it is connected to.

The $\mathbf{M}$ matrix we obtain from our example $\mathbf{C}$ matrix in (1) is

$$\begin{bmatrix} & \text{Bill} & \text{Bill G} & \text{G} & \text{cat} & \text{child} & \text{dog} & \text{fat} & \text{hat} & \text{in} & \text{lost} & \text{the} & \text{wild} \\ \text{Bill} & \mathbf{0} & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{Bill G} & 1 & \mathbf{0} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{G} & 1 & 1 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{cat} & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 1 & 0.5 & 1 & 0 & 0.38 & 0 \\ \text{child} & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0.5 & 0.12 & 0 \\ \text{dog} & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0.5 & 0.38 & 1 \\ \text{fat} & 0 & 0 & 0 & 0.33 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0.12 & 0 \\ \text{hat} & 0 & 0 & 0 & 0.33 & 0 & 0 & 0 & \mathbf{0} & 1 & 0 & 0.25 & 0 \\ \text{in} & 0 & 0 & 0 & 0.33 & 0 & 0 & 0 & 0.5 & \mathbf{0} & 0 & 0.12 & 0 \\ \text{lost} & 0 & 0 & 0 & 0 & 1 & 0.33 & 0 & 0 & 0 & \mathbf{0} & 0.25 & 0 \\ \text{the} & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{0} & 1 \\ \text{wild} & 0 & 0 & 0 & 0 & 0 & 0.33 & 0 & 0 & 0 & 0 & 0.12 & \mathbf{0} \end{bmatrix} \quad (3)$$

where entries are rounded to two decimal places for display purposes only.

At first, the loss of symmetry in $\mathbf{M}$ may seem counterintuitive. Consider that noun-article pairs are very common in English, so the probability of the word “the” occurring given that the word “cat” occurs is relatively high. As demonstrated in (3), the probability of “the” occurring in a chunk when “cat” occurs is $M(\text{“the”}, \text{“cat”}) = 1$ because, in Figure 2, “the” occurs in every chunk when “cat” occurs. However, the probability of “cat” occurring in a chunk when “the” occurs is $M(\text{“cat”}, \text{“the”}) = 0.38$ because “cat” does not

always occur in a chunk when “the” occurs. Figure 6 demonstrates what the graph represented by matrix $\mathbf{M}$ looks like for the small example given in Figure 2.

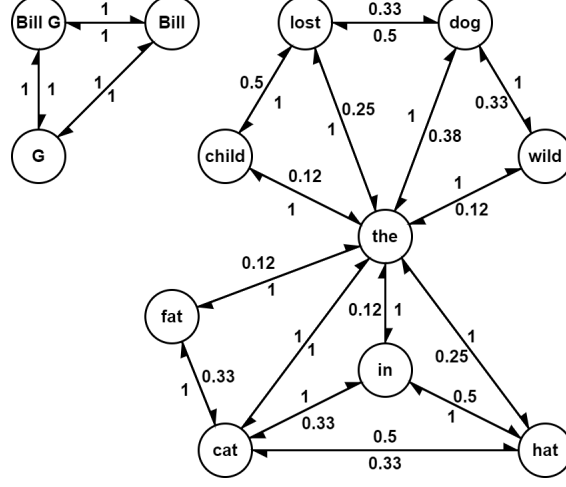


Figure 6: Graphical representation of the $\mathbf{M}$ matrix in (3). We note that directional pairs are merged into single lines with opposing arrows for visual simplicity.

3.4 The State Transition Probability Matrix

From the standardized matrix, $\mathbf{M}$, we construct the state transition probability matrix, $\mathbf{S}$. Entries in $\mathbf{S}$ are conditional probabilities defined as

$$S_{i,j} = \frac{M_{i,j}}{\sum_{k=1}^N M_{i,k}} \quad (4)$$

where $S_{i,j}$ represents the probability of transitioning from term i to term j in a single step through the graph. Using (4), we obtain $\mathbf{S}$ by row normalizing matrix $\mathbf{M}$. We choose to row normalize $\mathbf{S}$ because, when we calculate the expected hitting time, we want to traverse the graph from every other node to our root node.

The $\mathbf{S}$ matrix we obtain from our example $\mathbf{M}$ matrix in (3) is

$$\begin{bmatrix} & \text{Bill} & \text{Bill G} & \text{G} & \text{cat} & \text{child} & \text{dog} & \text{fat} & \text{hat} & \text{in} & \text{lost} & \text{the} & \text{wild} \\ \text{Bill} & \mathbf{0} & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{Bill G} & 0.5 & \mathbf{0} & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{G} & 0.5 & 0.5 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{cat} & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0.35 & 0.17 & 0.35 & 0 & 0.13 & 0 \\ \text{child} & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0.8 & 0.2 & 0 \\ \text{dog} & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0.27 & 0.2 & 0.53 \\ \text{fat} & 0 & 0 & 0 & 0.73 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0.27 & 0 \\ \text{hat} & 0 & 0 & 0 & 0.21 & 0 & 0 & 0 & \mathbf{0} & 0.63 & 0 & 0.16 & 0 \\ \text{in} & 0 & 0 & 0 & 0.35 & 0 & 0 & 0 & 0.52 & \mathbf{0} & 0 & 0.13 & 0 \\ \text{lost} & 0 & 0 & 0 & 0 & 0.63 & 0.21 & 0 & 0 & 0 & \mathbf{0} & 0.16 & 0 \\ \text{the} & 0 & 0 & 0 & 0.12 & 0.12 & 0.12 & 0.12 & 0.12 & 0.12 & 0.12 & \mathbf{0} & 0.12 \\ \text{wild} & 0 & 0 & 0 & 0 & 0 & 0.73 & 0 & 0 & 0 & 0 & 0.27 & \mathbf{0} \end{bmatrix} \quad (5)$$

where entries are rounded to two decimal places for display purposes only. Figure 7 represents the graph of the $\mathbf{S}$ matrix (5) for the small example given in Figure 2.

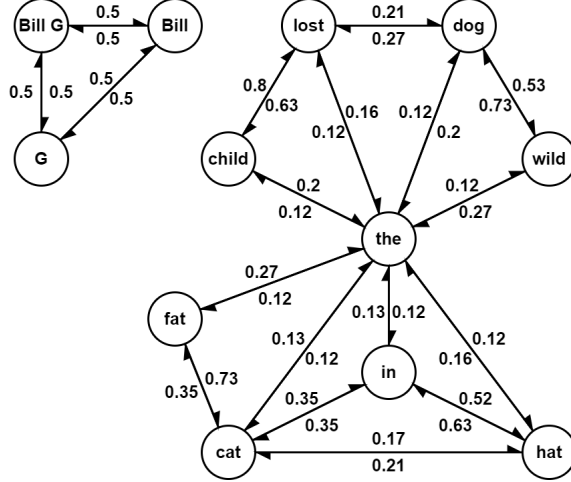


Figure 7: Graphical representation of the $\mathbf{S}$ matrix in (5).

3.5 The Approximated Model Matrix

While use of the full $\mathbf{S}$ matrix is ideal, in practice, approximations of $\mathbf{S}$ may be needed in order to accommodate the very large matrices that expansive training data sets can generate. One such approximation we consider is truncating matrix $\mathbf{S}$ by eliminating relatively low-probability edges to obtain an approximate matrix, $\mathbf{T}$.

We define $\mathbf{T}$ as the following:

Definition 1. For every node in $\mathbf{S}$, we keep the t out-edges with the highest values where t is a user-defined truncation threshold, $t > 0$. If a node has fewer than t edges, we keep all of its edges and its degree remains the same. Afterwards, we re-normalize the rows of matrix $\mathbf{T}$.

We note that setting t too low can remove high-probability edges and change the model’s behavior, resulting in substandard concepts. In general, for natural-language models containing around 10-50 million nodes and 1-10 billion edges, we use a threshold of $t = 1000$. For larger models, we find a lower threshold of around $t = 100$ may be necessary.

This approximation method drastically decreases the size of $\mathbf{S}$ while maintaining prominent edges that influence the Markov walk in the concept generation (Section 4.2.1). It also allows us to take fewer steps through the graph, further decreasing the computational time in the concept generation stage. We note that this approximation method does not reduce the matrix’s dimensions.

We define the concept generation calculations (Section 4) in terms of matrix $\mathbf{S}$.

4 Concept Generation

With our model, $\mathbf{S}$, we can generate concepts for a user’s chosen root term, provided that the root term exists as a node in $\mathbf{S}$. A concept is a weighted query expansion where every term in the model is assigned a weight that represents its relevance to the root term.

For a term in an MSR concept, we score its relevance to the root term in the following manner: if a concept term appears in a chunk of data, one can expect that the chunk is relevant to the concept’s root term. The higher the term’s weight, the more likely it is that the chunk containing the term will be relevant to the root term. We note that this scoring metric for relevance is not a direct similarity score between the term and the root term. Additionally, concepts are not symmetrical.

4.1 Overview

In general, we seek to find the terms that are likely to occur with our root term. Consider the following real-world example: if the term “golden retriever” appears in a chunk of text, it is extremely likely that

the chunk is relevant to the term “dog.” However, if the term “dog” appears in a chunk of text, it is not necessarily likely that the chunk is relevant to the term “golden retriever” because it could also be similarly relevant to the terms “German shepherd” or “poodle”. Therefore, creating a concept for the root term “dog” results in a very high weight for the term “golden retriever” but creating a concept for the root term “golden retriever” results in a noticeably lower weight for the term “dog.”

Term similarity approaches may accomplish this relevance ranking, but they also find undesired and common terms. To remove unwanted terms, we first find term distances and then perform a statistical comparison. We perform a Markovian graph traversal and calculate expected hitting times to measure distances for our root term with every model term. Instead of using these distance scores for the final concept weights, we also perform a statistical comparison of how our chosen root term’s distances to all model terms compare to the distributions of the distances for all pairs of model terms. For this statistical comparison, we use negated z-scores. Furthermore, we use a subset of graph nodes’ expected hitting time distances to approximate the distributions used in the statistical comparison. This greatly reduces the computational load and allows us to feasibly generate concepts from models created with very large training data sets.

4.2 Term Distance Calculation

In order to measure the initial distance scores for all model nodes to our chosen root node, we calculate the expected hitting time from each node to the root node. We note that we are not using the term *distance* to denote the formal mathematical distance that would define a metric space. We considered other term distances, such as Dijkstra’s shortest-path algorithm and PageRank, but determined the expected hitting time produced the best results in conjunction with the other aspects of MSR.

For all subsequent steps, we modify our model $\mathbf{S}$ such that it only contains the nodes that are in the same connected component as the root node. In the example $\mathbf{C}$ matrix defined in (1), we can see the nodes corresponding to “Bill”, “Bill G”, and “G” do not share edges with any other nodes in the graph and are therefore in a separate connected component from all of the other nodes. The number of connected components does not change during the transformations to the $\mathbf{M}$ and $\mathbf{S}$ matrices. Nodes that are in separate connected components from the root node are not be reachable when traversing the graph during node distance calculations (Section 4.2.1) and are therefore ignored.

4.2.1 Expected Hitting Time

For a chosen root node in the model, we calculate the expected hitting time from each node in the model to the root node. This results in a vector where each component is a distance score between the corresponding node and the root node. Rather than traversing the graph *from* the root node to other nodes, we traverse the graph *to* the root node from the other nodes.

We calculate the expected hitting time ${}^k v_\lambda^r$ from a given node k to the root node r in the model graph $\mathbf{S}$ by determining what fraction of Markovian random walks starting at node k end at node r after n Markovian steps through $\mathbf{S}$ for random walks with steps $n = 1, \dots, \lambda$. Let ${}^k \alpha_n^r$ be the probability that a random walk with n steps starting at node k arrived at the root node r on the n^{th} step. Let ${}^k \beta_n^r$ be the probability that a random walk with n steps starting at node k arrived at the root node r before or during the n^{th} step. As n approaches ∞ , ${}^k \alpha_n^r$ approaches 0 and ${}^k \beta_n^r$ approaches 1 because a random walk with $n = \infty$ steps starting at node k will eventually arrive at the root node r .

Let $\mathbf{S}$ be the model matrix of size $N \times N$, let k be the index of our starting node, and let r be the index of our root node. We define the basis vector $\mathbf{\Gamma}_i$ as a 1-hot encoding column vector of length N with a 1 in the i^{th} position and 0s elsewhere.

Before the first step through the graph, we initialize the column vector

$$\alpha_0^r = \mathbf{\Gamma}_r \tag{6}$$

and the scalar

$${}^k \beta_0^r = 0. \tag{7}$$

For steps $n \geq 1$,

$${}^k \alpha_n^r = \mathbf{\Gamma}_k \mathbf{S} \alpha_{n-1}^r \tag{8}$$

where $\mathbf{\Gamma}_k$ is a row vector and

$${}^k\beta_n^r = {}^k\beta_{n-1}^r + {}^k\alpha_n^r (1 - {}^k\beta_{n-1}^r). \quad (9)$$

Then, the expected hitting time from node k to the root node r after $n = 1, \dots, \lambda$ steps is

$${}^k v_\lambda^r = \sum_{n=1}^{\lambda} n ({}^k\alpha_n^r (1 - {}^k\beta_{n-1}^r)). \quad (10)$$

As n approaches ∞ , ${}^k v_n^r$ converges to the expected hitting time and can be used to measure the distance from node k to the root node r .

4.2.2 Expected Hitting Time Vector

For every step n in (8), we need to assemble the vector α_{n-1}^r by calculating the scalar ${}^k\alpha_{n-1}^r$ for each node k in $k = 1, \dots, N$. We modify our calculation for ${}^k\alpha_n^r$ in (8) by omitting the row vector $\mathbf{\Gamma}_k$. This now results in the vector α_n^r , which allows us to simultaneously calculate ${}^k\alpha_n^r$ for each node k in $k = 1, \dots, N$.

Now, we can convert our calculations to operate in the vector space. We begin by initializing the following column vectors of length N :

$$\alpha_0^r = \mathbf{\Gamma}_r \quad (11)$$

and

$$\beta_0^r = \mathbf{0}. \quad (12)$$

For steps $n \geq 1$, we have the column vectors

$$\alpha_n^r = \mathbf{S}\alpha_{n-1}^r \quad (13)$$

and

$$\beta_n^r = \beta_{n-1}^r + \alpha_n^r (1 - \beta_{n-1}^r) \quad (14)$$

where the multiplication of $\mathbf{S}$ and α_{n-1}^r is a matrix-vector product and the product $\alpha_n^r (1 - \beta_{n-1}^r)$ is a component-wise multiplication.

Then, the vector of expected hitting times from all model nodes to the root node after $n = 1, \dots, \lambda$ steps is

$$v_\lambda^r = \sum_{n=1}^{\lambda} n (\alpha_n^r (1 - \beta_{n-1}^r)). \quad (15)$$

4.2.3 Approximate Expected Hitting Time Vector

We approximate the convergence of each node's expected hitting time by assuming that a random walk from every node arrives at the root node during a random walk with $\lambda + 1$ steps:

$$\alpha_{\lambda+1}^r = \mathbf{1}. \quad (16)$$

Then, we accumulate the results of this random walk with $\lambda + 1$ steps:

$$\begin{aligned} v_{\lambda+1}^r &= v_\lambda^r + (\lambda + 1) (\alpha_{\lambda+1}^r (1 - \beta_\lambda^r)) \\ &= v_\lambda^r + (\lambda + 1) (1 - \beta_\lambda^r). \end{aligned} \quad (17)$$

Finally, we define our vector of expected hitting time distances from each node in $\mathbf{S}$ to the root node r :

$$v^r = v_\infty^r \approx v_{\lambda+1}^r. \quad (18)$$

Notably, taking more steps before this approximate step will result in a more accurate estimation for the expected hitting time's convergence. Moreover, when using the approximate model $\mathbf{T}$, it is best to still use the full model $\mathbf{S}$ when calculating α_n^r for the 1st step and then use $\mathbf{T}$ for all subsequent steps. This guarantees the traversal from immediately co-occurring nodes to the root node and helps combat traversal issues that may arise because of diminishing graph connectivity that the model approximation in Section 3.5 can introduce.

4.3 Statistical Comparison

We incorporate a statistical significance score that enables term filtering because term distances tend to produce high scores for common and somewhat similar terms. For the expected hitting time distance¹ ${}^k v^r$ from a given model node k to our chosen root node r , we assign its final concept weight by using a *negated* z-score to compare ${}^k v^r$ against the distribution of distances from node k to all model nodes.

Concept nodes with high negated z-scores are relevant to the root node. The graph nodes corresponding to these nodes have a low distances to the root node and higher distances to most other model nodes. Incorporating this statistical comparison metric for concept node weights enforces that more common nodes do not have high weights because, even though they have low distances to the root node, they also have low distances to other model nodes.

4.3.1 Z-Score Calculation

Let $\mathbf{v}^r$ be the column vector of length N containing the expected hitting time distances from all model nodes to our root node as obtained in Section 4.2.3. In principle, we assemble the expected hitting time column vectors for all model nodes j for $j = 1, \dots, N$ into matrix $\mathbf{X}$:

$$\mathbf{X} = [\mathbf{v}^1 \ \mathbf{v}^2 \ \dots \ \mathbf{v}^j \ \dots \ \mathbf{v}^N] \quad (19)$$

where each row $\mathbf{X}_k$ of $\mathbf{X}$ is the distribution of expected hitting times from node k to every node j .

To calculate the negated z-score for node k to the root node r , we compute the mean μ_k and standard deviation σ_k of row $\mathbf{X}_k$. Then, the negated z-score ${}^k z^r$ that compares the expected hitting time from node k to our root node r with the distribution of expected hitting times $\mathbf{X}_k$ from node k to all model nodes is

$${}^k z^r = - \left(\frac{{}^k v^r - \mu_k}{\sigma_k} \right). \quad (20)$$

With this, we assemble a vector of negated z-scores $\mathbf{z}^r$ by computing ${}^k z^r$ for all k in $k = 1, \dots, N$. This vector contains the final concept term weights that describe how each term k in the model relates to the root term, r .

4.3.2 Approximate Z-Score Distribution

Ideally, we would compute the expected hitting time vectors for every node in the model to use in our z-score distance comparison distributions, but this can become computationally impractical with larger models. Instead, we approximate these distributions by only calculating the expected hitting time vectors for a sample of reference nodes in the model.

We quickly select a set of reference nodes in an effort to find a representative approximation of the actual distribution by selecting a user-defined number of reference nodes Q that have the most similar number of edges, or degree, to the mean node degree of the model, $\mathbf{S}$. We do not incorporate edge weights for this determination. While there may be more precise methods for selecting reference nodes, we believe these nodes work well for the collection of approximate distributions that are used for comparison with any arbitrary root node's term distances. The means and standard deviations of these distributions can be precomputed and cached for the model, allowing efficient reuse with any chosen root term's term distance vector. Furthermore, this selection process is deterministic, giving us repeatable results. In practice, we find that $Q = 150$ is generally sufficient to approximate these distributions in natural-language models. The selection process for reference nodes is still an active area of research.

Now, we assemble our Q expected hitting time vectors into an $N \times Q$ matrix:

$$\mathbf{X}' = [\mathbf{v}^1 \mathbf{v}^2 \ \dots \ \mathbf{v}^Q]. \quad (21)$$

Since matrix $\mathbf{X}'$ uses relatively few nodes for its approximation, we find that it is necessary to remove outliers from its distributions. This removes problematic distance scores such as where reference nodes have

¹The term distance calculation can be substituted with an alternative calculation. We chose to use the the expected hitting time to measure term distances.

extremely close distances to each other or if the root node is in the reference distribution set. This removes nodes that are so frequent that they do not contain any useful information and this removes nodes whose distances are so large that they are unlikely to be relevant.

For every row $\mathbf{X}'_k$ of $\mathbf{X}'$, we calculate its mean μ_k . Then, for each element $X'_{k,j}$ for $j = 1, \dots, Q$, we calculate its distance $\delta_{k,j}$ from the row's mean, μ_k :

$$\delta_{k,j} = |X'_{k,j} - \mu_k|. \quad (22)$$

Let $p = \lceil R * Q \rceil$ be the number of elements to be removed from each row $\mathbf{X}_k$, where R is an arbitrary value in $[0, 1]$. We use $R = 0.2$ and suggest keeping $R < 0.5$. In each row $\mathbf{X}'_k$, we remove the p elements that have the largest distances $\delta_{k,j}$ from the row's mean, μ_k . Let $\mathbf{Y}$ be the resulting approximate matrix of size $N \times (Q - p)$; it can be used in place of matrix $\mathbf{X}$ as the matrix of expected hitting time distributions for the z-score calculations (Section 4.3.1).

4.4 Preparing Concepts for Applications

We define several metrics when presenting concepts. We define the **rank** as the position of a concept's term based on a sorting algorithm that involves the concept terms' weights and their co-occurrence counts with the root term (Section 4.4.1). The **weight** of a term is the relevance score obtained for that term resulting from the processes outlined in Sections 4.3 and 4.4.2. The **co-occur** is the number of chunks of training data where each term occurred with the root term. The **count** is the number of chunks of training data that contained the term. We introduce methods for scaling, cropping, and manipulating concepts, which enable users to more easily apply them to various applications.

4.4.1 Concept Term Sorting

At this stage, the MSR concept is an unordered vector of term-weight pairs, where the weights are the z-scores obtained in Section 4.3. The vector has as many term-weight pairs as there are nodes in the model. Most applications that use concepts are more efficient when the list of term-weight pairs is sorted and cropped to only include impactful items. For a variety of applications that use a cropped concept terms list, such as ranked, semantic search, we have determined that simply sorting the terms by their respective weights from highest to lowest is not the most ideal approach. Although, simply sorting and cropping concepts by their term weights works well for other applications, such as analyzing concept terms to discover subtle relationships.

Our sorting for concept terms incorporates the terms' co-occurrence counts with the root term and their z-score weights. First, we remove all terms with an arbitrary z-score weight < 6.0 because we consider any z-score weight above this weight threshold to be relevant to the root term. Then, we sort all remaining terms from highest to lowest based on their co-occurrence count with the root term. The co-occurrence count for root term r and term t is obtained from matrix $\mathbf{C}$ at position $C_{r,t}$. For terms that have the same co-occurrence count, we further sort them by their z-score weights from highest to lowest. Finally, we ensure that the root term is at the top of the list.

4.4.2 Scaling Concept Weights

Human comprehension is important for understanding how concept terms precisely relate to the root term with regards to their weights. We suggest performing an affine translation of the concept terms' weights in relation to the root term to make the weights more user-friendly. Let $\mathbf{y}$ be our concept terms' z-score weights as obtained in Section (4.3), let w_r be the root term's weight, and let a be the target weight of the root term. Then our scaled concept vector, $\mathbf{y}'$, is

$$\mathbf{y}' = \mathbf{y} / (w_r * a) \quad (23)$$

where these are component-wise operations.

4.4.3 Cropping Concepts

For natural language models with millions of terms, we suggest cropping the concept terms sorted by their rank to an arbitrary number of terms between 100 and 5000 terms.

4.4.4 Manipulating Concepts

Concepts are suggestions for generic applications; they may be modified to fit specific needs for applications by adding, removing, or editing terms and weights. We advise the user to ensure they do not introduce duplicate terms.

5 Applications

There are many applications for MSR concepts. We provide examples and analyses for four applications: analyzing concepts to find semantically related terms, using a multi-data-type model to discover subtle relationships across data types, cross-language query expansion with multi-lingual models, and using cross-language query expansion to perform ranked, semantic searches on independent, untranslated data at an arbitrary, sub-document level. The presented concepts are generated with 1000 Markovian expected hitting time steps through the model, 50 reference terms, a z-score weight threshold of 6.0, and an affine translation of the terms’ weights placing the root term’s weight at 100.0. The concepts are sorted by their rankings and cropped.

5.1 Discovery with an MSR Query Expansion

We present an example of an MSR concept created using a model trained on data from Simple Wikipedia (November 1, 2021 dump). Simple Wikipedia is an English-based version of the popular online encyclopedia, Wikipedia, designed with more accessible language. Simple Wikipedia articles tend to be shorter and contain a significantly smaller vocabulary of words than English Wikipedia articles, making it an excellent data set for testing because it produces smaller models than full English Wikipedia. In order to create the processed model training data, we segmented articles into chunks by the newline character and used the spaCy tokenizer with named entity recognition to extract terms.

For the example concept presented in Table 1, we chose the root term “Hermione Granger”, a character from the popular young adult book series *Harry Potter*, because relevant terms associated with the series tend to be easily recognizable.

Table 1 displays a cropped list of the highest ranked terms returned by MSR for the “Hermione Granger” concept generated using the Simple Wikipedia model. It showcases MSR’s ability to discover semantic relationships and promote subtly semantic relationships. The ordered relevance of these terms expresses both aspects of semantic relatedness: taxonomic relationships (semantic similarity) and thematic relationships. Furthermore, the MSR concept automatically punishes the weights for unrelated terms and common terms; when sorted by rank, concepts only include semantically related terms.

Upon meticulous examination, we see many terms that are specifically relevant to the root term “Hermione Granger” as well as the *Harry Potter* franchise including, but not limited to, “Harry Potter”, “Ron Weasley”, “Voldemort”, and “Hogwarts”. “Emma Watson” is the actress who portrayed “Hermione Granger” in the films. More subtle semantic relationships are also present. “September 19, 1979” is “Hermione Granger”’s birthday. “Alfonso Cuarón” and “Chris Columbus” directed *Harry Potter* films. “Quidditch”, “muggles”, “Slytherin”, “Ravenclaw”, and “Hufflepuff” are all less common terms specific to the *Harry Potter* franchise. “15 April 1990” is “Emma Watson”’s birthday and “Ballet Shoes” and “The Tale of Despereaux” are two unrelated films that “Emma Watson” starred in.

Arguably one of the most exciting features of MSR concepts is that they discover subtle relationships to terms that never co-occur with the root term in any chunk of the training data used to build the model. The following terms in Table 1 have a co-occur value of zero. “Kate DiCamillo” authored the novel *The Tale of Despereaux*, which was later produced as an animated film that “Emma Watson” starred in as a voice actor. “Ron Weasley”’s birthday is “24 August 1988” and “Rupert Alexander Lloyd Grint” is the full name

of the actor who portrayed him. “the Defense Against the Dark Arts” is a term specific to the *Harry Potter* franchise.

Rank	Weight	Co-Occur	Count	Term
1	100.0	28	28	Hermione Granger
2	45.42182	28	87	Hermione
3	40.94174	26	77	Granger
4	4.81879	23	1364	Harry
5	4.47023	18	478	Ron
6	19.36807	17	278	Harry Potter
7	12.78367	17	394	Potter
8	60.83552	16	57	Weasley
9	114.68555	15	22	Ron Weasley
10	36.56222	10	84	Voldemort
11	166.20739	9	14	Grint
12	166.20739	9	14	Rupert Grint
13	34.71999	9	85	Hogwarts
14	28.27217	9	26	Emma Watson
15	9.20843	9	130	Rupert
16	7.60775	9	246	magical
17	61.75513	8	23	Daniel Radcliffe
18	13.34399	8	81	Radcliffe
19	17.59217	7	97	Rowling
20	12.58524	7	131	wizard
...	...	...	...	...
29	18.62229	3	52	Severus
30	18.41431	3	21	the Chamber of Secrets
31	4.07734	3	74	Secrets
32	162.78902	2	3	Ronald Weasley
33	101.59513	2	2	September 19, 1979
34	91.55725	2	3	Sorting
35	86.9184	2	4	Hogwarts Express
36	74.00333	2	12	Harry Potter and the Philosopher's Stone
37	57.18486	2	5	Viktor Krum
38	52.16817	2	8	Quidditch
39	51.68601	2	7	muggles
40	50.33572	2	7	Rubeus
41	45.62383	2	9	Rubeus Hagrid
42	42.53741	2	6	Slytherin
43	39.28239	2	13	Ravenclaw
44	37.30311	2	8	Hufflepuff
45	36.34632	2	9	Hogwarts School of Witchcraft
46	34.99825	2	15	Hagrid
47	30.27033	2	17	Severus Snape
...	...	...	...	...
55	20.2615	2	13	Alfonso Cuarón
56	15.36206	2	15	Cuarón
57	11.93635	2	50	Witchcraft
58	10.88658	2	35	the Philosopher's Stone
59	9.62449	2	19	Chris Columbus
60	9.39943	2	44	Philosopher
61	4.89368	2	149	sorted
62	3.8252	2	138	brightest
63	304.30236	1	1	Severus Snape
64	304.30236	1	1	sixth year
65	216.16135	1	1	nonwizard
66	212.56576	1	1	Hermione Jean Granger
67	200.98159	1	4	Harry Potter and the Half-Blood Prince (movie)
68	196.60261	1	1	J K Rowling's
69	178.36603	1	1	Ballet Shoes
70	178.36603	1	1	As late as 2019
71	178.36603	1	1	15 April 1990
72	178.36603	1	1	Emma Charlotte Duerre Watson
73	178.36586	1	1	Duerre
74	166.21493	1	2	The Tale of Despereaux
...	...	...	...	...
249	3.71495	1	63	scar
250	3.6979	1	35	classmate
251	3.64202	1	26	Houten
252	3.64148	1	19	first year
253	114.76038	0	1	wheezes
254	113.90311	0	2	Rose Weasley
255	113.86526	0	2	Hugo Weasley
...	...	...	...	...
263	93.66959	0	4	Kate DiCamillo
264	93.66959	0	4	DiCamillo
265	88.06449	0	1	Rupert Alexander Lloyd Grint
266	83.0056	0	1	the Defense Against the Dark Arts
267	70.39274	0	2	24 August 1988
...	...	...	...	...

Table 1: Concept for the root term “Hermione Granger” created with the Simple Wikipedia model.

5.2 Query Expansion Across Data Types

This next concept example highlights the data type agnostic property of our MSR approach. We created a model that combines two disparate data sources— the same Simple Wikipedia data from Section 5.1 and Packet Capture (PCAP) data based on computer network activity. Our PCAP data set includes samples provided by contributors to the wiki for Wireshark [20], an open-source network data analysis tool, and Packet Life [16], a networking blog that includes a repository of PCAPs. To obtain the text-based representation of PCAP data necessary for MSR, we leveraged programmatic capabilities for processing network data and structured text. This portion of the model training data required some custom parsing to extract terms, including tokenization for sections containing HTML/XML from web pages. Most other terms were created as key-value pairs of common packet fields to better depict relationships between numeric values within network data, such as port numbers and protocol versions, that may otherwise lose context when performing MSR query expansion. An example of this is use of the term “tcp.srcport_80” rather than “80” to denote the Hypertext Transfer Protocol (HTTP) for web traffic.

As explained in Section 4, concepts can only be generated for root terms that exist in the model. This means concepts generated from multiple data sources must have the root term in at least one of the initial data sets. For the concept presented in Table 2, we chose the root term “Blue Mountains”, a mountain region in Australia, which appears in both the Simple Wikipedia and PCAP data sets. By using training data from these two sources, we are able to demonstrate MSR’s ability to discover semantic relatedness across multiple data types.

The results in Table 2 reveal many terms related to the root term “Blue Mountains” and the general Australian region, including the city “Sydney” and its beachside suburb “Narrabeen”. The terms “Lawson”, “Blaxland”, and “Wentworth” refer to towns in/near the Blue Mountains that are named for three explorers who discovered a passage from Sydney through the Blue Mountains— William Lawson, Gregory Blaxland, and William Wentworth. Some of the terms unique to the PCAP data set, such as “./2001-12-ReidarISydney/index.html” and “./2001-12-Reidar-BlueMountains/index.html”, refer to links within a web page containing various images and details, including a subset of Australian landmarks and attractions like the Blue Mountains. Further down in the table, there are also terms like “eth.src_00:c0:df:20:6c:df” and “ip.dst_10.1.1.101”, which can be used to identify specific network traffic related to accessing that web page.

Rank	Weight	Co-Occur	Count	Term
1	100.0	11	11	Blue Mountains
2	3.80606	11	1109	Blue
3	2.95211	11	1014	Mountains
4	40.04667	3	10	Australia Post
5	2.44737	3	109	postage
6	2.24062	3	104	Lawson
7	1.94542	3	138	stamp
8	1.64154	3	114	honoured
9	149.38444	2	5	Blue Mountains (New South Wales)
10	9.74671	2	17	Blaxland
11	6.4147	2	1267	Sydney
12	3.23713	2	71	Wentworth
13	325.58963	1	1	the City of Hawkesbury
14	325.58963	1	1	the City of Lithgow
15	325.58963	1	1	the City of Blue Mountains
16	186.72563	1	1	Udhagamandalam
17	186.72563	1	1	about 7,500 feet
18	138.93219	1	2	Ootacamund
19	124.61244	1	5	Blue Mountains tree frog
20	114.20213	1	1	Sydney Buses
21	114.20213	1	1	Inner West
22	114.20206	1	1	the Sutherland Shire
23	114.20206	1	1	the International and Domestic
24	99.96769	1	1	the end of the First Age
25	99.96769	1	1	the Gulf of Lh��n/Lune
...	...	...	...	...
86	93.05314	1	1	./2001-12-Reidar-BlueMountains/index.html
...	...	...	...	...
128	93.05314	1	1	./2001-12-ReidarISydney/index.html
...	...	...	...	...
143	93.05314	1	1	./2001-HusetNarrabeen/index.html
...	...	...	...	...
152	86.80889	1	8	Narrabeen
...	...	...	...	...
176	68.87453	1	17	SeaWorld
177	66.79503	1	204	ip.src_10.1.1.1
178	66.79503	1	204	ip.src_host_10.1.1.1
179	66.79503	1	204	eth.src_00:c0:df:20:6c:df
180	66.6468	1	7	att
181	63.62514	1	5	Central Coast
182	62.89822	1	11	liv
183	62.84696	1	20	Gosford
184	61.46176	1	18	som
185	57.8281	1	32	lite
186	57.798	1	4	Nilgiris
187	57.5078	1	10	frame.number_38
188	56.61287	1	13	nya
189	55.24582	1	277	ip.dst_10.1.1.101
190	55.24582	1	277	eth.dst_00:04:e2:22:5a:03
191	55.24582	1	277	ip.dst_host_10.1.1.101
...	...	...	...	...
224	19.2222	1	10	William Lawson (explorer)
241	9.91237	1	20	William Wentworth
243	8.98573	1	15	Gregory Blaxland
...	...	...	...	...

Table 2: Concept for the root term “Blue Mountains” created with the combined Simple Wikipedia and PCAP data model.

5.3 Cross-Language Query Expansion

In this example, we demonstrate MSR’s ability to cross the language barrier without performing any traditional machine translation. We trained a multi-lingual model on data from Chinese Wikipedia (November 1, 2021 dump) and a set of English-Chinese aligned bitext sentences. We concatenated each of the aligned English and Chinese sentence pairs and treated them as chunks.

We created a concept for the English root term “asteroid,” which appears in both training data sets. The results in Table 3 demonstrate that MSR concepts can produce cross-language query expansions using multi-lingual models. The second highest ranked term is the direct translation to the English root term, “asteroid”. The remaining terms contain a mixture of English and Chinese semantically related terms. The “Hayabusa” spacecraft landed on the asteroid “Itokawa”. “99942 阿波菲斯” and “2005YU55” are both named asteroids.

Rank	Weight	Co-Occur	Count	Term	English Google Translation
1	100.0	876	876	asteroid	asteroid
2	0.42178	613	33488	小行星	asteroid
3	0.41357	178	69713	地球	Earth
4	0.89455	58	1708	隕石	meteorite
5	1.80493	51	1893	近地	near the ground
6	2.98809	45	789	comet	comet
7	0.56316	42	8454	火星	Mars
8	1.54553	35	3223	spacecraft	spacecraft
9	0.62392	31	11655	sample	sample
10	1.70026	30	28195	surface	surface
11	3.71976	26	2629	collision	collision
12	0.79381	24	87582	enough	enough
13	11.19933	23	59	Hayabusa	Hayabusa
14	1.13724	23	1318	Jupiter	Jupiter
15	0.66642	23	7560	gravity	gravity
16	17.24588	18	31	Itokawa	Itokawa
17	1.50461	18	17871	scientists	scientists
18	0.60626	18	38433	hit	hit
19	1.46543	17	387	asteroids	asteroids
20	53.40466	16	102	多分派	multiple dispatch
21	13.50739	15	51	apophis	apophis
22	0.65911	15	880	rendezvous	rendezvous
23	0.47661	15	2519	太阳系	solar system
24	0.44068	15	756	神星	God Star
25	8.35146	14	58	隼鸟	Hayabusa
26	0.38514	14	78	星带	star belt
27	14.21544	12	757	spaceship	spaceship
28	2.73195	12	53	阿波菲斯	Apophis
29	1.97325	12	3668	hitting	hitting
30	0.56317	12	407	rearrange	rearrange
31	0.52421	12	16203	rock	rock
32	0.42794	12	37726	size	size
33	1.69858	11	5023	望远镜	telescope
34	0.62555	11	23280	sun	sun
35	1.57481	10	2567	infrared	infrared
...	...	...	...	...	...
321	2423.72837	1	1	带矿	with ore
322	2185.3384	1	1	臭隕石	smelly meteorite
...	...	...	...	...	...
468	30.19317	1	1	99942 阿波菲斯	99942 Apophis
...	...	...	...	...	...
623	10.89399	1	1	2005YU55	2005YU55
...	...	...	...	...	...

Table 3: Concept for the root term “asteroid” created with the multi-lingual Chinese Wikipedia and English-Chinese bitext model.

5.4 Ranked, Semantic Search of Data at an Arbitrary, Sub-document Level

Once the user has created a concept for their chosen root term, they can use MSR concept terms and weights to *semantically* search and prioritize data at an arbitrary, sub-document level. A key aspect to note is that any search data can be used with any concepts; it is not necessary to train a model and create subsequent concepts from the search data in order to semantically search the data.

5.4.1 Ranked, Semantic Search Approach

First, we crop the concept so it only has terms with positive, nonzero weights. If a chunk of search data contains one of the concept terms, the chunk is considered relevant and is assigned a score; all irrelevant chunks have a score of zero. The total relevance score for a given chunk is described as follows:

```
Data: Concept, SearchChunkList
Result: SearchChunkList
for SearchChunk in SearchChunkList do
  for (term, weight) in Concept do
    for term in SearchChunk do
      | SearchChunk.Score  $\leftarrow$  SearchChunk.Score + weight
    end
  end
end
```

where we search only on the set of terms in the chunk by excluding duplicates. Relevant search chunks are then sorted by their score. Those with the highest scores are the most semantically relevant to the user’s root term. For models and search data with natural language data, we found that cropping the concept to include only the top 10 to 100 ranked terms to be effective for semantic searches. When ranking entire documents, where each document contains a varying amount of chunks, we find it is better to individually score all of the document’s chunks and then assign the document’s score to its max scoring chunk instead of scoring the entire document as one chunk. However, this is still an active research topic.

5.4.2 Ranked, Semantic Search with Multiple Concepts

We implement a slightly different approach when a user wishes to search using multiple concepts. The concepts do not have to be created from the same model or with the same parameters. Before searching, we normalize their weights to have the same root term weights by performing affine translation on each concept as described in Section 4.4.2. For every chunk, we first search and rank the data with each concept individually. Each chunk is scored as above, where a chunk score is the sum of the set of the term weights. Next, we discard all chunks that are not relevant to any of the concepts. If a chunk contains no terms from a particular concept, the chunk’s score for that concept is zero. We then add a small $\epsilon \approx 0.000001$ to each individual chunk concept score to ensure they are nonzero and then take the geometric mean of the chunk’s individual concept scores. By scoring a chunk with respect to each concept and taking the geometric mean of the individual scores, we enforce the idea that, for a chunk to be useful to the user, it should be relevant to all of the root terms the user chose. This prevents search results that are dominated by a subset of the root terms.

5.4.3 Ranked, Cross-Language, Semantic Search on Untranslated, Independent Data

In this example, we perform a ranked, sub-document level semantic search using English search terms on a subset of untranslated, Chinese Common Crawl newswire containing 11 million chunks (paragraphs). Our query topic for this search data was to find Chinese newswire articles that were related to the topic of *asteroids endangering Earth*. We created three concepts from the same multi-lingual Chinese Wikipedia and English-Chinese bitext model from the previous example (Section 5.3) using the English root terms “asteroid”, “danger”, and “Earth”. These cropped concepts are shown in Table 4.

We semantically searched and ranked every chunk (11 million paragraphs) in the untranslated, Chinese newswire collection using only the top 10 ranked terms from each of the concepts shown in Table 4. It is important to note that we did not train the model using the search data; they are independent data sets.

Several of the highest scoring semantic search result chunks are displayed in Table 5 along with their English translations. As we can see, top results are all related to the query topic even though they do not contain the English root terms.

Rank	Weight	Co-Occur	Count	Term	English Google Translation
1	100.0	876	876	asteroid	asteroid
2	0.42178	613	33488	小行星	asteroid
3	0.41357	178	69713	地球	Earth
4	0.89455	58	1708	陨石	meteorite
5	1.80493	51	1893	近地	near the ground
6	2.69132	45	789	comet	comet
7	0.56316	42	8454	火星	Mars
8	1.54553	35	3223	spacecraft	spacecraft
9	0.62392	31	11655	sample	sample
10	1.70026	30	28195	surface	surface
1	100.0	23456	23456	danger	danger
2	20.28861	15595	107218	危险	Danger
3	2.16429	1017	32144	substantial	substantial
4	6.9801	969	5099	believing	believing
5	2.61293	383	4322	imminent	imminent
6	2.61619	351	3408	危险性	Dangerous
7	3.63564	301	240214	place	place
8	5.20434	262	6201	poses	poses
9	3.62021	208	2854	驱回	refoulement
10	48.82741	192	213	refouler	refouler
1	100.0	24532	24532	Earth	Earth
2	49.32796	18503	69713	地球	Earth
3	23.71159	2646	17101	观测	observation
4	23.831	1733	23477	satellite	satellite
5	56.2268	1691	17856	observation	observation
6	31.09241	1393	7433	orbit	orbit
7	29.9733	986	13374	planet	planet
8	14.7013	942	235841	global	global
9	11.59373	889	15987	atmosphere	atmosphere
10	10.95425	798	7872	遥感	remote sensing

Table 4: Concepts for the root terms “asteroid”, “danger”, and “Earth” created with the multi-lingual Chinese Wikipedia and English-Chinese bitext model.

Rank	Score	Terms Found	Chunk Text	English Translation
2	13.046	危险 (Danger) 地球 (Earth) 近地 (near the ground)	科技日报北京 12 月 30 日电据美国太空新闻网近日报道, 美国国家航空航天局 (NASA) 近地天体 (NEO) 新任务近日获得 3560 万美元的研发启动资金。这一任务名为“近地天体监视任务” (NEO Surveillance Mission), 将利用一台带有红外热像仪的小型太空望远镜发现和监视直径大约 140 米的近地天体, 以“揪出”其中对地球构成危险的天体。	Science and Technology Daily, Beijing, December 30. According to a recent report by Space News, NASA’s new Near-Earth Object (NEO) mission recently received \$35.6 million in R&D start-up funding. The mission, called the NEO Surveillance Mission, will use a small space telescope with a thermal imaging camera to find and monitor near-Earth objects about 140 meters in diameter to “spot out” the Dangerous celestial bodies to Earth.
3	12.008	危险 (Danger) 地球 (Earth) 小行星 (asteroid) 陨石 (meteorite)	- 小行星防御系统预警陨石来袭可计算是否处于危险中 (记者张梦然) 美国国家公共广播电台官网 1 日消息称, 一个“大块头”陨石近距离飞掠地球, 天文学家借助美国航空航天局 (NASA) 研发的新系统, 成功标记并算出其危险系数。2016-11-02 15:51:24	- Asteroid defense system warns of meteorites and can calculate whether they are in danger (Reporter Zhang Mengran) The official website of National Public Radio said on the 1st that a “big” meteorite flew by the earth at close range, and astronomers used the help of NASA.) developed a new system that successfully marked and calculated its risk factor. 2016-11-02 15:51:24
5	9.42	危险 (Danger) 地球 (Earth) 小行星 (asteroid)	按照美国航天局的定义, 与地球距离在 7 4 8 万公里之内、直径超过 1 4 0 米的小行星, 才被认为对地球构成“潜在危险”。	According to the definition of NASA, only asteroids with a distance of 7.48 million kilometers from the earth and a diameter of more than 140 meters are considered to be “potentially dangerous” to the earth.
6	9.42	危险 (Danger) 地球 (Earth) 小行星 (asteroid)	对于地球来说, 近地小行星是一种危险的存在。它们在太空中风驰电掣, 不时与地球擦肩而过, 让人心惊肉跳。不过这些鲁莽的家伙并非一无是处, 它们身上也藏有科学家们梦寐以求的东西, 比如太阳系最古老的遗迹。	For Earth, near-Earth asteroids are a dangerous existence. They are galloping through space, passing by the Earth from time to time, which is heart-stopping. But these reckless fellows are not without merit, and they also harbor things scientists can only dream of, such as the oldest remains of the solar system.

Table 5: Several of the highest scoring semantic search result chunks for searching a subset of untranslated, Chinese Common Crawl newswire containing 11 million chunks (paragraphs) using concepts for the root terms “asteroid”, “danger”, and “Earth” created with the multi-lingual Chinese Wikipedia and English-Chinese bitext model.

6 Open Research Questions

MSR is still an active area of research. In this section, we discuss several open questions related to our work, including the selection of appropriate reference terms during the z-score calculation.

6.1 Alternative Ways to Calculate Reference Nodes and Concept Weights

At the moment, we obtain reference nodes in the process as outlined in Section 4.3.2. However, we considered several possibilities. For example, there may be an optimal number of reference nodes to be obtained based on the distribution of degrees of nodes in the graph. Another possibility may be to randomly sample reference nodes. In general, we seek a way to calculate reference nodes at the model creation stage to decrease the calculation time required for concepts.

In addition to modifying the way we choose reference nodes, we consider an alternate approach for using these nodes to calculate the required z-scores for concept weights. Currently, we assemble the expected hitting time vectors into the matrix $\mathbf{X}$ defined in (4.3.2). We then calculate the column means and remove the $p = 0.2 * Q$ outliers. This process is intended to remove irrelevant terms from the concept rankings, so we may then use high z-scores as a measure of high relevance. Further research is needed to determine whether this process is necessary for obtaining quality results, and if so, what value of p should be used.

References

- [1] Afuan, L., Ashari, A., and Suyanto, Y. (2021). A New Approach in Query Expansion Methods for Improving Information Retrieval. *JUITA: Jurnal Informatika*, 9(1), 93-103.
- [2] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, and M., and Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches.
- [3] Agirre, E., Arregi, X., and Otegi, A. (2010, August). Document expansion based on WordNet for robust IR. In *Coling 2010: Posters* (pp. 9-17).
- [4] Ataeva, O., Serebryakov, V., and Tuchkova, N. (2020). Query expansion method application for searching in masemantical subject domains. In *CEUR Workshop Proceedings* (pp. 38-48).
- [5] Bai, J., Song, D., Bruza, P., Nie, J. Y., and Cao, G. (2005, October). Query expansion using term relationships in language models for information retrieval. In *Proceedings of the 14th ACM international conference on Information and knowledge management* (pp. 688-695).
- [6] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135-146.
- [7] Cao, G., Gao, J., Nie, J. Y., and Bai, J. (2007, November). Extending query translation to cross-language query expansion with markov chain models. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (pp. 351-360).
- [8] Collins-Thompson, K., and Callan, J. (2005, October). Query expansion using random walk models. In *Proceedings of the 14th ACM international conference on Information and knowledge management* (pp. 704-711).
- [9] Dai, Z., and Callan, J. (2019, July). Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 985-988).
- [10] Kacmajor, M., and Kelleher, J. D. (2020). Capturing and measuring thematic relatedness. *Language Resources and Evaluation*, 54(3), 645-682.
- [11] Kosloski, J. T. and Thompson, J. W. (2017). *Method and device for measuring word pair relevancy* (U.S. Patent No. 9754020). U.S. Patent and Trademark Office.
- [12] Kosloski, J. T. and Thompson, J. W. (2019). *Method and device for measuring relevancy of a document to a keyword(s)* (U.S. Patent No. 10242090). U.S. Patent and Trademark Office.
- [13] Lavrenko, V., and Croft, W. B. (2017, August). Relevance-based language models. In *ACM SIGIR Forum* (Vol. 51, No. 2, pp. 260-267). New York, NY, USA: ACM.
- [14] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [15] Naseri, S., Dalton, J., Yates, A., and Allan, J. (2021). CEQE: Contextualized Embeddings for Query Expansion. *arXiv preprint arXiv:2103.05256*.
- [16] *Packet Captures*. (2009). Packet Life. Retrieved July 15, 2021, from <https://packetlife.net/captures/>
- [17] Padaki, R., Dai, Z., and Callan, J. (2020). Rethinking Query Expansion for BERT Reranking. *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*, 12036, 297-304.
- [18] Pedersen, T., Pakhomov, S. V., Patwardhan, S., and Chute, C. G. (2007). Measures of semantic similarity and relatedness in the biomedical domain. *Journal of biomedical informatics*, 40(3), 288-299.

- [19] Pilehvar, M. T., Jurgens, D., and Navigli, R. (2013, August). Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1341-1351)
- [20] *SampleCaptures*. (2020, August 11). Wireshark Wiki. Retrieved October 1, 2021, from <https://wiki.wireshark.org/SampleCaptures>
- [21] Sankhavara, J. (2020). Feature weighting in finding feedback documents for query expansion in biomedical document retrieval. *SN Computer Science*, 1(2), 1-7.
- [22] Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4), 35-43.
- [23] Zheng, Z., Hui, K., He, B., Han, X., Sun, L., and Yates, A. (2020, November). BERT-QE: Contextualized Query Expansion for Document Re-ranking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings* (pp. 4718-4728).