An Unsupervised Query Rewriting Approach Using N-gram Co-occurrence Statistics to Find Similar Phrases in Large Text Corpora

Hans Moen[1], **Laura-Maria Peltonen**[2], **Henry Suhonen**[2,3], **Hanna-Maria Matinolli**[2], **Riitta Mieronkoski**[2], **Kirsi Telen**[2], **Kirsi Terho**[2,3], **Tapio Salakoski**[1] **and Sanna Salanterä**[2,3]
[1]Turku NLP Group, Department of Future Technologies, University of Turku, Finland
[2]Department of Nursing Science, University of Turku, Finland
[3]Turku University Hospital, Finland
{hanmoe,lmemur,hajsuh,hmkmat,
ritemi,kikrte,kmterh,sala,sansala}@utu.fi

Abstract

We present our work towards developing a system that should find, in a large text corpus, contiguous phrases expressing similar meaning as a query phrase of arbitrary length. Depending on the use case, this task can be seen as a form of (phrase-level) query rewriting. The suggested approach works in a generative manner, is unsupervised and uses a combination of a semantic word n-gram model, a statistical language model and a document search engine. A central component is a distributional semantic model containing word n-grams vectors (or embeddings) which models semantic similarities between n-grams of different order. As data we use a large corpus of PubMed abstracts. The presented experiment is based on manual evaluation of extracted phrases for arbitrary queries provided by a group of evaluators. The results indicate that the proposed approach is promising and that the use of distributional semantic models trained with uni-, bi- and trigrams seems to work better than a more traditional unigram model.

1 Introduction

When searching to see if some information is found in a text corpus, it may be difficult to formulate search queries that precisely match all relevant formulations expressing the same information. This becomes particularly difficult when the information is expressed using multiple words, as a phrase, due to the expressibility and complexity of natural language. Single words may have several synonyms, or near synonyms, which refer to the same or similar underlying concept (e.g. “school” vs “gymnasium”). When it comes to multi-word phrases and expressions, possible variations in word use, word count and word order complicate things further (e.g. “consume food” vs “food and eating” or “DM II” vs “type 2 diabetes mellitus”).

An important task for a search engine is to try to bridge the gap between user queries and how associated phrases of similar meaning (semantics) are written in the targeted text. In this paper we present our work towards enabling phrase-level query rewriting in an unsupervised manner. Here we explore a relatively simple generative approach, implemented as a prototype (search) system. The task of the system is, given a query phrase as input, generate and suggest as output contiguous candidate phrases from the targeted corpus that each express similar meaning as the query. These phrases, input and output, may be of any length (word count), and not necessarily known as such before the system is presented with the query. Ideally, all unique phrases with similar meaning as the query should be identified. For example, the query might be: “organizational characteristics of older people care”. This exact query phrase may or may not occur in the target corpus. Regardless, a phrase candidate of related meaning that we want our system to identify in the targeted corpus could then be: “community care of elderly”. In this example, the main challenges that we are faced with are: 1) how can we identify these four words as a relevant phrase, and 2) decide that its meaning is similar to that of the query. Depending on the use case, the task can be seen as a form of query rewriting/substitution, paraphrasing or a restricted type of query expansion. Relevant use cases include information retrieval, information extraction, question–answering and text summarization. We also aim to use this functionality to support manual annotation. For that purpose the system will be tasked with finding phrases that have similar meaning as exemplar phrases and queries provided by the user, and/or as previously annotated text spans. An unsupervised approach like we are aiming for would be particularly valuable for corpora and in domains that lack relevant labeled training data, e.g. in the form of search history logs, needed for supervised paraphrasing and query rewriting approaches.

The presented system relies on a combination of primarily three components: A distributional semantic model of word n-gram vectors (or embeddings), containing unigrams, bigrams and trigrams; A statistical language model; And a document search engine. Briefly explained, the way the system works is by first generating a set of plausible phrase (rewrite) candidates for a given query. This is done by first composing vector representation(s) of the query, and then searching for and retrieving n-grams that are close by in the semantic vector space. These n-grams are then concatenated to form the phrase candidates. In this process, the statistical language model helps to quickly discard phrases that are likely nonsensical. Next the phrases are ranked according to their similarity to the query, and finally the search engine checks which phrase candidates actually exist in the targeted corpus, and where.

Similar to Zhao et al. (2017) and Gupta et al. (2019) we explore the inclusion of word n-grams of different sizes in the same semantic space/model. One motivation for this is that they both found this to produce improved unigram representations compared to only training with unigram co-occurrence statistics. Another motivation is that we want to use the model to not only retrieve unigrams that are semantically close to each other, but also bigrams and trigrams.

2 Related Work

Unsupervised methods for capturing and modeling word-level semantics as vectors, or embeddings, have been popular since the introduction of Latent Semantic Analysis (LSA) (Deerwester et al., 1990) around the beginning of the 1990s. Such word vector representations, where the underlying training heuristic is typically based on the distributional hypothesis (Harris, 1954), usually with some form of dimension reduction, have shown to capture word similarity (synonymy and relatedness) and analogy (see e.g. Agirre et al. (2009); Mikolov et al. (2013)). Methods and toolkits like Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) are nowadays commonly used to (pre-)train word embeddings for further use in various NLP tasks, including supervised text classification with neural networks. However, recent methods such as ELMo (Peters et al., 2017) and BERT (Devlin et al., 2018) use deep neural networks to represent context sensitive word embeddings, which achieves state-of-the-art performance when used in supervised text classification and similar.

Further, there are several relatively recent works focusing on using and/or representing n-gram information as semantic vectors (see e.g. Bojanowski et al. (2016); Zhao et al. (2017); Poliak et al. (2017); Gupta et al. (2019)), possibly to further represent clauses, sentences and/or documents (see e.g. Le and Mikolov (2014); Pagliardini et al. (2018)) in semantic vector spaces.

A relatively straight forward approach to identify and represent common phrases as vectors in a semantic space is to first use some type of collocation detection. Here the aim is to identify sequences of words that co-occur more often than what is expected by chance in a large corpus. One can then train a semantic model where identified phrases are treated as individual tokens, on the same level as words, like it is done in Mikolov et al. (2013).

In the works mentioned so far, the focus is on distributional semantics for representing and calculating semantic similarity and relatedness between predefined lexical units and/or of predefined length (words/n-grams, collocations, clauses, sentences, etc.). Dinu and Baroni (2014) and Turney (2014) take things a step further and approach the more complex and challenging task of using semantic models to enable phrase generation. Their aim is similar to ours: given an input query (phrase) consisting of k words, generate as output t phrases consisting of l words that each expresses its meaning. Their approaches rely on applying a set of separately trained vector composition and decomposition functions able to compose a single vector from a vector pair, or decompose a vector back into estimates of its constituent vectors, possibly in the semantic space of another domain or language.

Dinu and Baroni (2014) also apply vector composition and decomposition in a recursive manner for longer phrases ($t \leq 3$). Their focus is on mapping between unigrams, bigrams and trigrams. As output their system produce one vector per word which represent the (to be) generated phrase. Here the evaluation primarily assumes that $t = 1$, i.e. the nearest neighbouring word in the semantic model, belonging to the expected word class, is extracted per vector to form the output phrase. However, no solution is presented for when $t > 1$ other than independent ranked lists of semantically similar words to each vector.

Turney (2014) explores an approach targeting retrieval of multiple phrases for a single query (i.e. $t > 1$), evaluated on unigram to bigram and bigram to unigram extraction. Here he applies a supervised ranking algorithm to rank the generated output candidates. For each input query, the evaluation checks whether or not the correct/expected output (phrase) is among the list of top hundred candidates.

It is unclear how well these two latter approaches potentially scale beyond bigrams or trigrams. Further, they assume that the length of the input/output phrases is known in advance. However, the task that we are aiming for is to develop a system that can take any query phrase of arbitrary (sub-sentence) length as input. As output it should suggest phrases that it identifies in a large document corpus which express the same or similar information/meaning. Here the idea is that we only apply upper and lower thresholds when it comes to the length of the output phrase suggestions. In addition, we do not want to be concerned with knowledge about word classes in the input and output phrases. We are not aware of previous work presenting a solution to this task.

In the next section, Section 3, we describe how our system works. In Section 4 we present a preliminary evaluation followed by discussion and plans for future work directions.

3 Methods

3.1 Semantic Model Training

In order to train a semantic n-gram model of unigrams, bigrams and trigrams, we initially explored two approaches. First using the Word2Vecf (Levy and Goldberg, 2014) variation of the original Word2Vec toolkit, where one can freely customize the word-to-context training instances as individual rows in the training file – each row containing one source word and one target context to predict. We opted for a skip-gram representation of the training corpus, meaning, for each row in the customized training file, we put the source n-gram and one of its neighboring n-grams as target context. The size of the sliding window is decided by how many neighboring (context) n-grams we include for each source n-gram. Overlap between the source n-gram and target n-grams is allowed.

However, we found that Word2Vecf only allows training using negative sampling. As an alternative approach we simply used the original Word2Vec toolkit, with the skip-gram architecture, hierarchical softmax optimization and a window size of one, to train on the same word-to-context organized training file intended for Word2Vecf. This means that it sees and trains on only two n-grams (cf. word–context pair) at a time. Based on preliminary testing we found this latter approach to produce semantic models that seemed to best capture n-gram semantics for our use case.

The text used for training the semantic model is first stemmed using the Snowball stemmer. This is done to normalize inflected word forms, reduce the number of unique n-grams and consequently the size of the model, as well as creating more training examples for the remaining n-grams. Mapping back to full-form words and phrases is later done using a document search engine, as explained below.

3.2 Phrase-Level Query Rewriting System

Our system works in a generative way when trying to find phrases from a target corpus that are semantically similar to a query phrase. We describe this as a five-step process/pipeline.

Step 1: As a first step we generate a set of query vectors for each of the different n-gram orders in the model – uni, bi and tri. We simply generate these vectors by normalizing and summing the associated n-gram vectors from the semantic model. In addition, if a word (or all words in a n-grams when n > 1) is found in a stopword list[1], we give these vectors half weight. As an example: given the query "this is a query", we generate three query vectors, $\overrightarrow{q_{1\text{-}g}}$, $\overrightarrow{q_{2\text{-}g}}$ and $\overrightarrow{q_{3\text{-}g}}$ as follows:

$$\overrightarrow{q_{1\text{-}g}} = sum(\frac{1}{2}\overrightarrow{\text{this}}, \frac{1}{2}\overrightarrow{\text{is}}, \frac{1}{2}\overrightarrow{\text{a}}, \overrightarrow{\text{query}}) \quad (1)$$

$$\overrightarrow{q_{2\text{-}g}} = sum(\frac{1}{2}\overrightarrow{\text{this\_is}}, \frac{1}{2}\overrightarrow{\text{is\_a}}, \overrightarrow{\text{a\_query}}) \quad (2)$$

$$\overrightarrow{q_{3\text{-}g}} = sum(\frac{1}{2}\overrightarrow{\text{this\_is\_a}}, \overrightarrow{\text{is\_a\_query}}) \quad (3)$$

[1]We use the NLTK (Bird et al., 2009) stopword list for English.

If, let's say, the query only contains one word, we can not generate query bigram or trigram vectors. Also, not all n-grams might be found in the semantic model. To compensate for this possibility, we keep track of the *coverage percentage* of each composed vector. This is later used when calculating similarity between the query and the generated phrase candidates (see step 4).

Step 2: Having composed the query vectors, the second step focuses on using the semantic model to extract the most similar n-grams. For each query vector, $\overrightarrow{q_{1\text{-}g}}$, $\overrightarrow{q_{2\text{-}g}}$ and $\overrightarrow{q_{3\text{-}g}}$, we extract semantically similar unigrams, bigrams and trigrams that are near in the semantic space. As a distance measure we apply the commonly used cosine similarity measure (cos). We use a cut-off threshold and a max count as parameters to limit the number of retrieved n-grams and further the number of generated phrase candidates in step 3.

Step 3: The third step focuses on generating candidate phrases from the extracted n-grams. This is primarily done by simply exploring all possible permutations of the extracted n-grams. Here we apply the statistical language model, trained using the KenLM toolkit (Heafield, 2011), to efficiently and iteratively check if nonsensical candidate phrases are being generated. For n-grams where n > 1 we also combine with overlapping words – one overlapping word for bigrams and one or two overlapping words for trigrams. As an example, from the bigrams: "a_good" and "good_cake", we can construct the phrase "a good cake" since "good" is overlapping.

The generation of a phrase will end if no additional n-grams can be added, or if the length reaches a maximum word count threshold relative to the length of the query[2]. If, at this point, a phrase has a length that is below a minimum length threshold[3], it will be discarded. Finally, we also conduct some simple rule-based trimming of candidates by mainly removing stopwords if they occur as the rightmost word(s).

Step 4: After having generated a set of candidate phrases, we now rank these by their similarity to the query. For each phrase candidate we compose phrase vectors ($\overrightarrow{p_{n\text{-}g}}$) in the same way as we did for the query. That said, we observed that the trigram coverage of the semantic model is relatively low compared to unigrams and bigrams. This is a result of us using a minimum n-gram occurrence count threshold of 20 when training the semantic model. Thus, for the presented experiment, we decided to exclude trigrams in the similarity scoring function.

As already mentioned, not all n-grams may be found in the semantic model. Thus, we also incorporate what we refer to as coverage information for each $\overrightarrow{q_{n\text{-}g}} - \overrightarrow{p_{n\text{-}g}}$ pair. The underlying intuition is to let query vectors and phrase candidate vectors with low model coverage have a lower influence on the overall similarity score. For example, if phrase p is "this is a phrase", which consist of three bigrams, but the semantic model is missing the bigram "a_phrase", the coverage of $\overrightarrow{p_{2\text{-}g}}$, i.e. $cov(\overrightarrow{p_{2\text{-}g}})$, becomes $2/3 = 0.66$. The coverage of a $\overrightarrow{q_{n\text{-}g}}$ – $\overrightarrow{p_{n\text{-}g}}$ pair is simply the product of their coverage, i.e. $cov(\overrightarrow{q_{n\text{-}g}}) \times cov(\overrightarrow{p_{n\text{-}g}})$.

The overall similarity function $sim(q,p)$ for a query q and a phrase candidate p is as follows:

$$sim(q,p) = \frac{1}{cov_{sum}} \sum_{n=1}^{2} \sum_{m=1}^{2} \Big(cos(\overrightarrow{q_{n\text{-}g}}, \overrightarrow{p_{m\text{-}g}}) \times cov(\overrightarrow{q_{n\text{-}g}}) \times cov(\overrightarrow{p_{m\text{-}g}}) \Big) \quad (4)$$

Where cos is cosine similarity, $cov(\overrightarrow{q_{n\text{-}g}})$ and $cov(\overrightarrow{p_{m\text{-}g}})$ refer to their coverage in the semantic model, and cov_{sum} is:

$$cov_{sum} = \sum_{n=1}^{2} \sum_{m=1}^{2} \Big(cov(\overrightarrow{q_{n\text{-}g}}) \times cov(\overrightarrow{p_{m\text{-}g}}) \Big) \quad (5)$$

Finally, all candidate phrases generated from a query are ranked in descending order.

Step 5: In the final step we filter out the candidate phrases that are not found in the targeted text corpus. To do this we have made the corpus searchable by indexing it with the Apache Solr search platform[4]. Since the candidate phrases are at this point stemmed, we use a text field analyzer that allows matching of stemmed words and phrases with inflected versions found in the original text corpus. In this step we also gain information about how many times each phrase occur in the corpus, and where. By starting with the most similar candidate phrase, the search engine is used to filter out non-matching phrases until the

[2] max_length = query_length + 2 `if` query_length ≤ 2 `else` query_length $\times$ 1.50

[3] min_length = 1 `if` query_length ≤ 2 `else` query_length $\times$ 0.50

[4] `http://lucene.apache.org/solr`

desired number of existing phrases are found, or until there are no more candidates left.

In addition, the system checks to see if an exact match of the query exist in the corpus. If this is the case, it removes any phrase candidate that are either a subphrase of the query or contains the entire query as a subphrase. This is a rather strict restriction, but for evaluation purposes it ensures that the system does not simply find and suggest entries of the original query phrase (with some additional words), or subphrases of it.

4 Experiment

Evaluating the performance of such a system is challenging due to the complexity of the task and the size of the text corpus. We are not aware of evaluation data containing gold standards for this task. Also, the complexity of the task makes it difficult to apply suitable baseline methods to compare against.

We decided to conduct a relatively small experiment, relying on manual evaluation, with the aim of getting an insight into strengths and weaknesses of the system. As text corpus we use a collection of PubMed abstracts consisting of approximately 3.6B tokens. Since our approach is unsupervised, we use this same data set for both training and testing. Six people (aka evaluators) with background as researchers and practitioners in the field of medicine were asked to provide 10 phrases of arbitrarily length, relevant to their research interests. The requirements were that the phrases should be intended for PubMed, more or less grammatically correct, and preferably consist of two or more words. This resulted in 69[5] phrases of different topics, length and complexity, with an average word count of 4.07. These serve as *query phrases*, or simply *queries*, for the remainder of this experiment.

Next, we use three different versions of the system, `Ngram`, `Unigram` and $\texttt{Ngram}_{restr}$ (described below) to separately generate and suggest 20 candidate phrases for each query. The evaluators were then given the task of assessing/rating if these phrases expressed the same information, similar information, topical relatedness or were unrelated to the query. Each evaluator assessed the suggestions for the query phrases they provided themselves[6]. The five-class scale used for rating

[5]One person submitted 19 phrases.

[6]No overlapping evaluation were conducted, so no inter-rater agreement information is available.

Class	Description
1	Same information as the query.
2	Same information as the query and has additional information.
3	Similar information as the query but is missing some information.
4	Different information than the query but concerns the same topic.
5	Not related to the query.

Table 1: Classes used by the evaluators when rating phrases suggested by the system.

the phrase suggestions can be seen in Table 1. In total, 1380 phrases were assessed for each system (69×20).

System - `Ngram`: Here the system is employed as it is described in Section 3.2. We prepared the training data for the semantic n-gram model with a window size equivalent to 3. Minimum occurrence count for inclusion was 20[7]. A dimensionality of 200 was used and otherwise default hyper parameters.

System - `Unigram`: Here we use a more traditional semantic model containing only unigrams. We trained the model using Word2Vec with skip-gram architecture, a dimensionality of 200, window size of 3, minimum inclusion threshold of 20, and otherwise default hyper parameters. This model was used to both extract relevant words and to calculate similarity between phrases and the query. Comparing this to the `Ngram` variant should provide some insight into the effect of training/using semantic models with word n-grams.

System - $\texttt{Ngram}_{restr}$: Here we add an additional restriction to the default setup (`Ngram`) by removing any generated phrase candidates containing one or more bigrams found in the query (based on their stemmed versions). The intention is to see if the system is still able to find phrases of related information to a query, despite not allowed to use any word pairs found in it.

In all system versions we use a statistical language model (KenLM (Heafield, 2011)) trained on the mentioned text corpus with an order of 3. We set the phrase inclusion likelihood threshold to

[7]Unique unigrams = 0.8M, bigrams = 6.5M, trigrams = 15.7M.

Class	System		
	`Ngram`	**`Unigram`**	**`Ngram`**$_{restr}$
1	13.99%	9.78%	8.48%
2	17.61%	12.54%	16.30%
3	24.13%	23.04%	16.23%
4	22.61%	25.14%	24.06%
5	21.67%	29.49%	34.93%
1+2	31.59%	22.32%	24.78%
1+2+3+4	78.33%	70.51%	65.07%

Table 2: Manual evaluation results.

-11.2. We strived to select parameters that made the system variants produce, on average, approximately the same number of phrase candidates (step 2 and 3). The number of phrase candidates generated in step 3 varied significantly depending on the query and system, from some thousands to some tens of thousands.

5 Results, Discussion and Future Work

Table 2 shows how the evaluators rated the (rewrite) phrases extracted by the various system setups. With the `Ngram` variant, when allowed to suggest 20 phrases, 31.59% of these contain the same information as the query phrases – possibly with some additional information (rated class 1+2). 78.33% of the suggested phrases concerns the same topic as the query phrases, i.e. rated class 1+2+3+4. The latter indicate the percentage of phrases that could be relevant to the user when it comes to query-based searching. Overall the results show that the system is indeed capable of generating, finding and suggesting (from the PubMed abstracts corpus) phrases that expresses similar meaning as the query. Table 3 shows examples of a few queries, rewrite suggestions by the system and their ratings by the evaluators.

Using a semantic model trained on word n-grams of different orders simultaneously (`Ngram`) achieves better results than using a unigram model (`Unigram`). This supports the findings in Zhao et al. (2017) and Gupta et al. (2019).

Naturally, the restricted `Ngram`$_{restr}$ variant achieves lower scores than `Ngram`. However, the performance differences are not that great when looking at the percentage of phrases rated as class 2. This suggests that the system finds phrases containing some additional information and/or phrases with words and expressions describing other degrees of specificity. Further, despite not allowed to suggest phrases containing bigrams found in the associated queries, it still achieves a higher 1+2 score than `Unigram`.

For some expressions used in the queries, there might not exist any good alternatives. Or, these might not exist in the PubMed abstracts corpus. For example, given the query “hand hygiene in hospitals”. Since `Ngram`$_{restr}$ is not allowed to suggest phrases containing the expression “hand hygiene”, or even “hygiene in”, it has instead found and suggested some phrases containing somewhat related concepts such as “hand-washing” and “hand disinfection”. However, for other queries the system had an arguably easier time. For example, for the query “digestive tract surgery” it suggests phrases like “gastrointestinal (GI) tract operations” (rated as class 1) and “gastrointestinal tract reconstruction” (rated as class 2). In other cases, the same meaning of a phrase is more or less retained when simply changing the word order (e.g. “nurses’ information needs” vs “information nurses need”).

We observed that step 5 typically took less time to complete for `Ngram` compared to `Unigram`. This could indicate that `Ngram` – using the n-gram model – is better at producing phrases that are likely to exist in the corpus. Another factor here is the effect of using the n-gram model in the ranking step (step 4), which retains some word order information from the queries.

A weakness of the conducted experiment is that we do not have a true gold standard reflecting if there actually exist any phrases in the corpus of similar meaning to the queries, or how many there potentially are. Still, the results show that the proposed system is indeed able to generate and suggest phrases whose information expresses the same or similar meaning as the provided queries, also when there are no exact matches of the query in the corpus. A planned next step is to look into other evaluation options. One option is to create a gold standard for a set of predefined queries using a smaller corpus. However, it can be difficult to manually identify and decide which phrases are relevant to a given query. Another option is to use the system to search for concepts and entities that has a finite set of known textual variants – e.g. use one variant as query and see if it can find the others. Alternatively, an extrinsic evaluation approach would be to have people use the system in an interactive way for tasks related to phrase-level

Query phrase	Rewrite suggestions by the system	Rating
infection prevention and control in hospital	• prevent and control hospital infections	1
	• control and prevent nosocomial infection	2
	• infection control and preventative care	4
information system impact	• information system influence	1
	• impact of healthcare information systems	2
	• health information system : effects	2
attitude and hand hygiene	• knowledge and attitude towards hand hygiene	2
	• Hand Hygiene : Knowledge and Attitudes	2
	• handwashing practices and attitudes	3
assessment of functional capacity of older people	• the functional assessment of elderly people	1
	• functional capacity of the elderly	3
	• the functional status of elderly individuals	4
facial muscle electromyography	• electromyography of facial muscles	1
	• electromyography (EMG) of masticatory muscles	2
	• facial muscle recording	3
treatment of post-operative nausea and vomiting	• postoperative nausea and vomiting (PONV) treatment	1
	• control of postoperative nausea and vomiting (PONV)	1
	• treatment of emesis , nausea and vomiting	4
fundamental care	• fundamental nursing care	2
	• palliative care is fundamental	2
	• holistic care , spiritual care	4
pain after cardiac surgery	• postoperative pain after heart surgery	1
	• postoperative pain management after cardiac surgery	2
	• discomfort after cardiac surgery	4

Table 3: Examples of a few queries, rewrite suggestions by the system and their ratings by the evaluators.

searching and matching, and then collect qualitative and/or quantitative feedback regarding impact on task effectiveness.

So far, not much focus has been placed on system optimization. For example, no multithreading was used in the phrase generation steps. The average time it took for the system to generate and find 20 phrases in the PubMed abstracts corpus for a query was about 30 seconds. This varied quite a bit depending on the number of n-grams extracted in step 2, the semantic model used and the length of the query. One bottleneck seems to be step 5 which is dependent on the size and status of the document index. However, it is worth noting that we have observed this to take only a few seconds for smaller corpora. For use in search scenarios where response time is critical, offline generation for common queries is an option. Further, this could for example serve to produce training data for supervised approaches.

As future work, system optimization will aim towards having the system generate as few non-relevant phrase candidates as possible while avoiding leaving out relevant ones. This includes making the search in the vector space (semantic model) to be as precise as possible (query vector composition) with a wide enough search for semantically similar n-grams (*cos* similarity cutoff threshold). Also, the similarity measure used to rank the phrase candidates relative to the query ($sim(q, p)$) is important for the performance of the system. As future work we also plan to look into the possibility of incorporating ways to automatically exclude non-relevant phrase candidates, e.g. by using a similarity cut-off threshold. Other text similarity measures and approaches could be tried, such as some of those shown to perform well in the SemEval STS shared tasks (Cer et al., 2017). In our relatively straight forward vector composition approach, each word/n-gram are weighted equally (except for stopwords). Improvements may be gained by incorporating some sort of statistical word weighting, like TF-IDF (Salton and Buckley, 1988). Other vector composition approaches could also be considered. Further, we also plan to explore other approaches to generating semantic text representations, such as Sent2Vec (Pagliardini et al., 2018). Also approaches like ELMo (Peters et al., 2017) and BERT (Devlin et al., 2018) could be applicable for this purpose. Additionally, one could also explore the use of cross-lingual semantic models for tasks related to translation.

Some times the system had a hard time finding phrases reflecting all the information in some of the more lengthy and complex queries – possibly referring to multiple topics. For example, “means to reduce the duration of surgical operations” and “a systematic approach to infection control”. For some of the queries one can assume that no contiguous (sub-sentence) phrases exist among the PubMed abstracts that expresses the same meaning. However, something that is missing from our current pipeline is some kind of query segmentation step. We are now treating each query as a single expression. As future work, especially in the context of query-based free-text searching, we aim to incorporate some sort of query segmentation which may split the query into smaller parts dependent on its complexity and the number of topics it refers to. Here we also want to explore the possibility of having wildcards in the query.

Overall we find these initial results to be promising. Further exploration and evaluation of the presented approach and system is needed. This includes looking into potential improvements and extensions, such as those mentioned above.

6 Conclusion

In this paper we have described a prototype system intended for the task of finding, in a large text corpus, contiguous phrases with similar meaning as a query of arbitrary length. For each of the 69 queries provided by a group of evaluators, we tested the system at finding 20 phrases expressing similar information. As corpus a large collection of PubMed abstracts were used. The results indicate that using a semantic model trained on word n-grams of different orders (1–3) simultaneously is beneficial compared to using a more traditional word unigram model. Further, when restricting the system from suggesting phrases containing bigrams from the corresponding queries, the results indicate that the system is still able to find and suggest relevant phrases.

Acknowledgments

This research was supported by the Academy of Finland (315376). We would like to thank Suwisa Kaewphan for helping out with the preprocessing of the PubMed abstracts data set.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: NAACL 2009*, pages 19–27. Association for Computational Linguistics.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. OReilly Media Inc., Sebastopol, California, USA.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada. Association for Computational Linguistics.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Georgiana Dinu and Marco Baroni. 2014. How to make words with vectors: Phrase generation in distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 624–633.

Prakhar Gupta, Matteo Pagliardini, and Martin Jaggi. 2019. Better word embeddings by disentangling contextual n-gram information. *arXiv preprint arXiv:1904.05033*.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10:146–162.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT ’11, pages 187–197, Stroudsburg, PA, USA. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–540. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765.

Adam Poliak, Pushpendre Rastogi, M Patrick Martin, and Benjamin Van Durme. 2017. Efficient, compositional, order-sensitive n-gram embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 503–508.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *IPM*, 24(5):513–523.

Peter D Turney. 2014. Semantic composition and decomposition: From recognition to generation. *arXiv preprint arXiv:1405.7908*.

Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 244–253.