2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00242

Semantic Search Pipeline: From Query Expansion to Concept Forging

Elizabeth Soper[1,2], Jordan Hosier[1,3], Dustin Bales[1], and Vijay K. Gurbani[1,4]

[1]Vail Systems, Inc.
[2]Department of Linguistics, State University of New York at Buffalo
[3]Department of Linguistics, Northwestern University
[4]Department of Computer Science, Illinois Institute of Technology

***Abstract*—When searching a database for a topic (e.g. Covid-19), there may not exist a precise match, especially if the topic is novel. Furthermore, the topic may surface in the data under different guises ('Covid-19,' 'coronavirus,' 'pandemic', etc.). The results of a keyword search are limited by the querier's imagination and familiarity with the data. Such searches have high precision, but low recall. In order to increase the recall of searches, we present the Semantic Search Pipeline, a novel approach to document retrieval that uses distributional semantic models and locality sensitive hashing to expand queries and efficiently identify other relevant documents that may not contain the obvious query terms. We evaluate the pipeline using a dataset curated from financial customer service call centers, resulting in an increase in recall of 32% over a simple keyword baseline, with a negligible drop in precision. Furthermore, we present the notion of *concept forging*, a process of tracing a topic or concept through time and through its various surface realizations. Applied to Covid-19, the search pipeline retrieves a set of documents that allow us to uncover the short- and long-term effects of Covid-19 on the lives of the people and businesses impacted by it. Although Covid-19 is a timely test case, our search pipeline is general in nature and can be easily applied to any range of topics.**

I. Introduction

Despite impressive advances in search efficiency across large datasets [1]–[5], document retrieval remains a challenging problem in natural language processing (NLP). Identifying documents relevant to a query topic often requires deep domain knowledge to craft appropriate queries, and even this may not be enough to extract all relevant information. With a simple keyword search, finding what you're looking for can be a challenge. Often a querier has a general idea of the topic they are interested in, but may not know how this topic surfaces in the data. The keywords that the querier comes up with may leave out many relevant documents and consequently, the search results may not be truly representative of the data. On the other hand, more sophisticated search methods which use state-of-the-art deep learning models may yield better results, but are prohibitively costly in terms of time and computational resources.

To retrieve a more comprehensive set of relevant results than a simple keyword search, it is important to find a balance between sophistication and efficiency. To this end, we employ distributional semantic models to access the underlying meaning of keywords and of unstructured text documents. The novelty of our method lies in using distributional semantic models trained at two levels, word and document, while minimizing computational cost with locality sensitive hashing, to create a fuller picture of how the query topic shows up across the entire dataset.

The pipeline described in this report was created for mining a database of call transcripts from financial customer service call centers. It contains approximately 100K call transcripts per day over the course of many months, transcribed by an automatic speech recognition system. The rest of the paper is organized as follows: Section II summarizes relevant previous work on query expansion and document retrieval, Section III outlines the steps in the pipeline, in Section IV we evaluate its performance on a test query, in Section V we discuss how the results of the pipeline can be used for *concept forging* to provide insight into the query topic, and finally we conclude in Section VI.

II. Related Work

Distributional semantics is based on the intuition that "you shall know a word by the company it keeps" [6]; that is, the meaning of a word can be learned from the contexts where it occurs in language. Words are represented as vectors, and words with similar meaning having correspondingly similar vector representations. Distributional semantic models have been successfully applied to a variety of NLP tasks, including information retrieval; some previous work has used word-level embeddings for query expansion [1]–[3], while Le and Mikolov (2014) [7] create document-level embeddings to model overall similarity between two documents.

Locality sensitive hashing (LSH) [8] is an efficient approximation of nearest neighbor search. Hashing is a technique that maps data to a fixed-size value, and is used to decrease the time needed to retrieve individual records. Generally, the goal of a hash function is to minimize the chance that two unique data points will hash to the same value (or collide). LSH differs from conventional hashing in that these collisions are maximized instead of minimized; similar items are more likely to hash into the same 'bins' than into different ones. Since similar items end up in the same bins, locality sensitive hashing can be used to approximate a nearest neighbor search by checking only items in a single bin instead of every

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00242

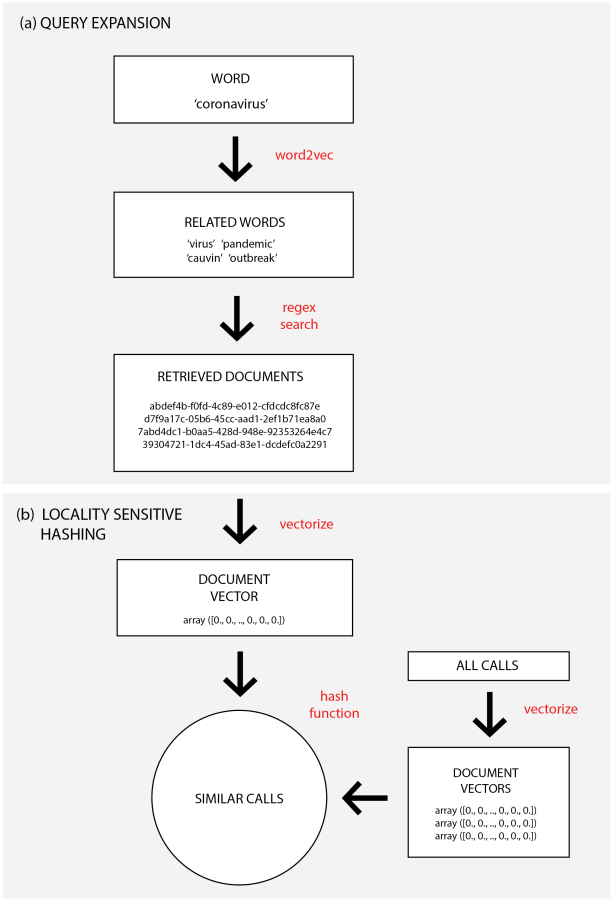

Fig. 1. An overview of the Semantic Search Pipeline

document in the dataset. LSH has been used on non-linguistic data [9] as well as linguistic data, especially for document retrieval [4], [5].

In previous work using LSH on text data, documents were represented with ngram 'shingles' or TF-IDF vectors, not distributional semantic models [4], [5]. Our work builds on this research by combining the advantages of distributional semantics with the efficiency of LSH. Using distributional models at both word and document level gives better results, while maintaining a sublinear query runtime.

III. Semantic Search Pipeline

The Semantic Search Pipeline is designed to take one or more keywords as input and return a list of relevant documents from the database as output. The pipeline has two major components: query expansion (using word2vec) and near neighbor search (using LSH). Fig. 1 illustrates the process.

A. Query Expansion with Word2vec

The first component of the Semantic Search Pipeline takes the input query terms, retrieves an additional list of terms based on a word2vec model, and returns the results of a keyword search based on the expanded query. This is illustrated in Fig. 1(a). Word2vec is a type of distributional semantic model which uses a feedforward neural network to learn vector representations of words based on a large text corpus. In a trained model, words with similar meaning will have similar vector representations [10]. Word vector models have been used successfully in the past for query expansion; this step in the Semantic Search Pipeline is very similar to the approaches used in [1]–[3].

Based on a word2vec model, a similarity threshold t, a list of query terms Q, and the model's vocabulary V, the system returns any item in V whose cosine similarity to one of the query terms is at least t (Eq. 1 below).

$$\forall q \in Q, \forall v \in V, (v|cos(v,q) \geq t) \tag{1}$$

B. Near Neighbor Search with Locality Sensitive Hashing

Even with an expanded query, some relevant results are not captured. Sometimes the reference is implicit. For example, in the phrase 'everything going on right now' (from a transcript in April 2020), it's clear that the caller is referring to Covid-19, even though there are no search terms in the phrase. These references will be clear to a human reader, but difficult to define in terms of keywords. To retrieve some of these more elusive cases where the query topic is implicit in the document, we again turn to vector representations. However, we now use vector representation at the document level to find other documents similar to those retrieved by the expanded query. Following the same logic as the first component of the pipeline, related documents will have similar vectors. Our intuition is that finding the nearest neighbours of documents we know are relevant can help identify additional documents that are also relevant but do not contain any of the specified keywords.

Because our dataset is large (approximately 54M documents), calculating the similarity between the query and every document in the dataset is impractical as it will exhibit linear complexity. While we can bound the time complexity by performing similarity calculations in parallel on multiple cores and machines, it is nonetheless of interest to seek approximate solutions through randomized algorithms that exhibit sublinear time. To do so, we opt for an efficient approximation of a full nearest neighbor search by using LSH.

The second component of the pipeline is illustrated in Fig. 1(b). In this component, each document in the dataset is mapped to a binary vector of length 10. Formally, let $D = \{D_1, D_2, ..., D_n\}$, where $\vec{D_i}$ is a vector representation of document i, and n is the total number of documents. We seek to map $\vec{D_i}$ to a binary vector of some fixed length, i.e.,

$$\forall i \in \{1..n\} \quad \vec{D_i} \mapsto \vec{\mathcal{H}} \tag{2}$$

Our locality sensitive hash, $\vec{\mathcal{H}}$, is a combination of 10 hash functions, $h_i,\ i = \{1..10\}$. $\vec{\mathcal{H}}$ is defined in Eq. 3 as follows:

$$\vec{\mathcal{H}} = \bigcup_{i=1}^{10} h_i \tag{3}$$

where each function, h_i, corresponds to a random hyperplane that helps bin vector $\vec{D_i}$. The hash function, h_i, is defined as[1]:

$$h_i(\vec{D_i}) = \begin{cases} 1 & \text{if } \vec{D_i} \text{ falls below the hyperplane} \\ 0 & \text{if } \vec{D_i} \text{ falls above the hyperplane} \end{cases} \quad (4)$$

In this manner, each document in our dataset is mapped to a binary vector of length 10 (Eq. 2). Consequently, documents will hash into one of 2^{10} possible bins (i.e., unique 10-digit sequences of 1's and 0's). Because similar vectors are more likely to have the same hash output, similar documents are, therefore, more likely to land in the same bin.

Based on the keyword search from the first step, we create a 'proto-document' by averaging the vectors for each document that contains at least one word *v* found by Eq. 1. Since all of these documents are known to be relevant to the search topic, averaging their vectors amplifies the patterns among them, while cancelling out document-specific noise. The resulting 'proto-document' vector is therefore a better approximation of the original query topic than any individual document vector. This averaged vector is then hashed with same function described above. All documents that hashed into same bin as this averaged vector are considered candidates. Any of these candidates which are similar enough to the averaged vector (given a set threshold) are returned. Thus, we can approximate an exhaustive neighbor search with fewer computations by comparing only documents which ended up in the same bin as our query, instead of the entire dataset.

IV. Evaluation

A. Dataset

The following evaluation is based a subset of the entire database, specifically the first 10K calls from March 27, 2020. Each document was a call transcript from a financial customer service call center. The data is multi-channel; each transcript consists of both the customer's and the employee's speech. This subset of 10K was small enough that the results could be manually verified.

B. Query Expansion Component

Evaluation of the query expansion component of the pipeline is based on word2vec models trained on 50K calls from April 1, 2020, and a similarity threshold of 0.8. This threshold was found to optimally balance the number of keywords returned with the relevance of the keywords, based on a grid search, where the search results were manually verified. Using word2vec to expand our test query resulted in a 30.4% increase in retrieved documents. The precision of this method is presumed to be approximately 100%, based on inspection of the expanded query terms.

Two word2vec models were trained and tested. The first was a typical unigram model, where each word in the corpus vocabulary has its own unique vector. The second model was constructed by identifying meaningful bigrams to include in the model's vocabulary, in addition to the unigram vocabulary of the first model. Meaningful bigrams were identified using pointwise mutual information (PMI), a measure of association between events. Formally, the PMI of two words x and y in a corpus is defined as follows:

$$pmi(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} \quad (5)$$

Pairs of words that occur together more than they occur with other words have high PMI, while words that are more weakly associated have low PMI. Based on this metric, bigrams with a PMI ≥ 10 were selected and added to the existing unigram vocabulary. During training, these bigrams were treated as distinct vocabulary items and with unique vector representations. For example, 'laid_off' has a vector that is distinct from the vector for 'laid' and the vector for 'off.' Table I shows the related terms identified by both the unigram and bigram models. Using a basic unigram word2vec model to expand query results in a 26.7% increase in retrieved documents from the original query ('coronavirus corona virus'). Adding bigrams to the model results in an additional 3.7% increase in retrieved documents, for a total 30.4% increase over the original unexpanded query (Table II).

Using word2vec to find additional query terms was especially useful for this particular dataset, which consists of call transcripts created through automatic speech recognition (ASR). With word2vec we quickly identified many ways the ASR system had mistranscribed *Covid* (a novel term which was not in the training set of ASR engines, and consequently never occurred in the transcribed data). The multitude of ways in which an ASR engine could mistranscribe *Covid*

TABLE I
SELECTED QUERY TERMS FOR EACH EXPANSION METHOD

Method	Keywords
Original Query	coronavirus, corona, virus
Expanded Query (basic word2vec)	kobe, krona, coronavirus, shutdown, colvard, pandemic, virus, outbreak, coronaviruses, coveted, epidemic, corona, koran, crisis, illness
Expanded Query (bigram word2vec)	coronavirus, colvard, coronaviruses, rotavirus, epidemic, current_situation, shutdown, virus, colvard_19, coveted_19, koven_19, virus_outbreak, crisis, cauvin, October_19, corona_virus, corona, cold_virus, kobe_19, pandemic, outbreak, cobra_19, whole_coronavirus, koran

TABLE II
EFFECT OF QUERY ON RECALL

Query Type	Retrieved Documents
Original query	1309
Expanded (basic word2vec)	1659
Expanded (bigram word2vec)	1707

[1]Our hash was implemented using Python's NearPy package: http://github.com/pixelogik/NearPy

TABLE III
EFFECT OF VECTOR FORMAT ON NEAR NEIGHBOR RESULTS

Vector Format	Precision	Novelty
TF-IDF	0.60	0.25
doc2vec	0.93	0.37

would be nearly impossible for a querier to guess without extensive examination of the data, but were easily identified using word2vec.

C. Near Neighbor Search Component

Evaluation of the pipeline's second component is based on two metrics: precision and novelty. Precision represents the percent of the documents returned that were relevant to the query. We define *novelty* as the percent of the documents returned that were both relevant and had not been retrieved previously by the keyword search. Because the query vector is the average of the documents retrieved in the first step, many of the nearest neighbors retrieved by LSH turn out to be documents retrieved by the first step. The usefulness of LSH is mainly in its finding new documents, so novelty is an important measure. In the rest of this section, we report the results of 1) testing different document vectorization methods and 2) tuning the LSH distance parameter, then 3) describe an optimization step in which we use random sampling to reduce the cost of creating a 'proto vector.'

1) Testing vector input formats: Two methods of vectorizing documents were tested. The first was a TF-IDF model [11]. In this representation, each dimension represents a word in the vocabulary. The value of each dimension is determined by the frequency of the word in the document (TF = term frequency), compared with its frequency in the overall dataset (IDF = inverse document frequency). If a word occurs many times in a document but rarely occurs in other documents, it is assumed to be important for classifying the document. The TF-IDF model was trained on the entire 10K call test corpus, excluding words that occurred less than 10 times or more than 500K times, resulting in a vocabulary of 6,472 words.

The second vector model tested was doc2vec [7]. This model is comparable to word2vec, except vectors can represent texts of variable length. Unlike the TF-IDF vectors, each dimension's meaning in doc2vec representations is opaque – it represents some pattern discovered by the neural network during training. The doc2vec vectors tested were 300 dimensions, trained on 100k calls from April 15, 2020.

Table III shows the precision and novelty rates for both formats, based on manually verifying the 30 nearest neighbors returned for the 'proto-document' vector in each format. Doc2vec representations performed the best, with 93% precision and a 37% novelty rate.

These are far from the only possible ways to represent documents as vectors. For example, representations based on transformer models, like BERT [12], have recently become the state-of-the-art in deep NLP. While using a model like this as input to our LSH function is possible in theory, in practice this defeats our purpose for using LSH in the first place: reducing computational cost. Encoding each document in a vector is the bottleneck step in the pipeline; the time required to compute one document vector was approximately 0.03s.[2] Since the size of BERT and similar deep learning models would increase the time and resources needed to perform a search, our doc2vec representations are a suitable compromise.

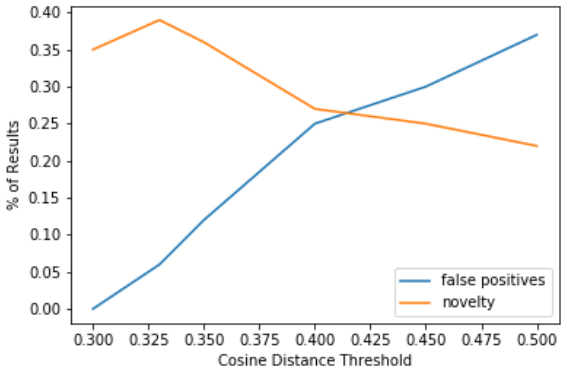

Fig. 2. Effect of Distance Threshold on False Positive and Novelty Rate

2) Testing distance threshold: The distance threshold parameter determines which, of all the candidate documents identified by LSH, will be returned. Distance is measured as the cosine between the 'proto' vector (created by averaging the results of keyword search) and the candidate vector (i.e. the document in the same hash bin being compared to the 'proto' vector). There is a trade-off between novel examples and false positives; the higher the the threshold, the more likely novel examples will be returned, but unrelated documents will also be more likely. A grid search was done on a range of thresholds to find the optimal value, which maximizes recall and minimizes false positives. Fig. 2 shows the results of this test. The optimal distance threshold was found to be between 0.30 and 0.35 (cosine distance).

The final version of our pipeline was able to successfully identify many Covid-19-related documents which did not contain any explicit reference to any of the expanded query terms. Fig. 3 shows just a few examples of Covid-19 references returned by the pipeline which would be nearly impossible to find using just keywords. We can see that the pipeline was able to identify documents which mention 'the current conditions' and 'the current emergency situation' as relevant to Covid-19. Even more remarkable is the fourth example, where the caller mentions they are a 'hairstylist;' the implication is that the caller has been negatively affected by Covid-19, since the hair salon industry was among those most affected by lockdowns in 2020. The pipeline was able to pick up on this implication, and identified the document as relevant.

Overall, the second component of our pipeline increased recall by an additional 1.5% (a total increase of 31.9% over the original keyword search). While fewer documents

[2]Time measurements based on an Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz with 40GB RAM.

...I was just wondering if you guys were doing any options for deferring p ayments given **what's going on right now in the economy in the world** ...

...I usually pay my bill in store with the stores all being closed with **ev erything going on**...

...And then I got laid off because of the **whole cronut thing** and I was jus t calling different places to see if we can get a couple months deferred p ossibly payment...

...I'm just checking with you because I'm a **hairstylist** and I don't get pa id right now due to **the closing**...

...Well I've been laid off because of the **current conditions everybody's u nder**...

...I don't know if the Postal Service is experiencing some delays due to t he **the current emergency situation**...

Fig. 3. Examples of implicit coronavirus references found by near neighbor search (emphasis added)

cauvin	cold_virus	kobe_19	koran
colvard	epidemic	coveted_19	koven_19
cobra_19	colvard_19	rotavirus	cova_19
october_19	kogut_19	covered_nineteen	krona
corbitt_19	cova	cauvin_19	cosied
kobe	koven	cold_midnineteenth	covered_19
cobia_19	koran_virus	kohver_19	colvert_19
rhinovirus	kogut	carone	coveted
culvert_19	crovitz	coba_19	co_bid
cove_at	cold_19	corvette_19	midnineteenth
clone	growing_virus	corporate_19	coded_19
kovac_19	coroner	covert	coalgate

Fig. 4. Mistranscriptions of 'coronavirus' or 'Covid-19'

were retrieved by the near neighbor search component than by query expansion, they add significant value in terms of representativeness of the entire corpus.

V. Concept Forging: Semantic Search as a Tool for Data Analysis

The results of the Semantic Search Pipeline facilitate new insight into our data and allow us to explore new directions in gathering insights about the query topic; the pipeline, in essence, allows us to observe how a concept relates to other concepts and how it evolves over time. This process, which we call *concept forging*, happens at multiple stages in the search process, based both on the distributional semantic models and on the collection of documents retrieved by the pipeline.

Simply looking at the word2vec neighbors of the query terms sheds some light on the query topic and its use in the corpus. This query expansion component of the pipeline is especially useful for ASR transcribed data, as it highlights transcription errors. Fig. 4 gives a list of ways that 'coronavirus' or 'Covid-19' were mistranscribed, based on inspection of their 100 nearest neighbors in the trained word2vec model. These mistranscriptions would be nearly impossible to guess when crafting a keyword search by hand, but can be quickly identified using a word2vec model trained on our data. Looking at slightly more distant neighbors of the query terms (generally between 0.6 and 0.8 cosine similarity) also points to some related concepts ('laid_off', 'shut_down', 'economy'), giving a broader picture of how the concept of interest (Covid-19) is connected to other concepts (e.g. employment).

Word2vec models are also a valuable source of insight into how concepts change over time. Several recent studies [13]–[16] have shown that training distributional models on different

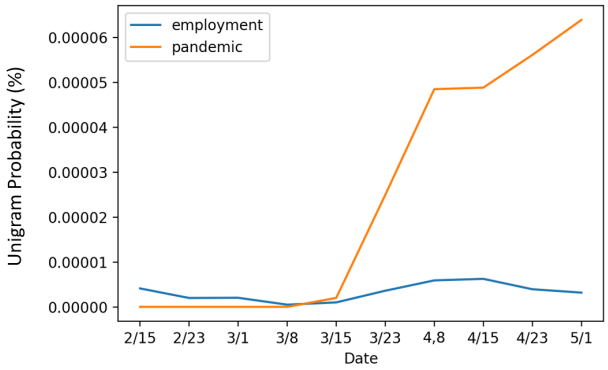

Fig. 5. Changes in frequency over time for 'employment' and 'pandemic'

time periods can illuminate trends in the data that help us understand how a word or concept evolves. This is more effective than looking only at changes in word frequency. For example, while the word 'pandemic' increases dramatically in frequency between January 2020 and April 2020, reflecting the arrival of Covid-19 in the US, the word 'employment' does not (Fig. 5). Despite the fact that it does not change in frequency, though, the contexts it is used in do change – it is used more often in relation to the coronavirus pandemic. This change in usage can be seen by looking at the word's nearest neighbors in the word2vec models pre-Covid and post-Covid; Fig. 6 illustrates how the meaning of a word can shift over time when it is used in different contexts. In January, before the coronavirus had hit the US, 'employment' is relatively far away from words like 'virus,' 'outbreak,' and 'corona.'[3] By April, however, all these words converge, reflecting the change in relationship between these terms as the pandemic caused many to lose their jobs. Thus we are able to identify a change in the concept of 'employment' that is not evident from frequency information alone.

Tredici et al. [17] suggest that in modeling short-term meaning shift, it is difficult to distinguish contextual changes caused by repeated reference to specific events from more robust semantic change, as a shorter time period is more likely to be influenced by individual events. We measured change over an even shorter time than Tredici et al. [17], looking at just a few months. We are not claiming, however, in our analysis of the change in 'employment,' that the word has fundamentally changed its meaning between January and April 2020. Rather, the shift in semantic space reflects its new (and likely temporary) association with the Covid-19 pandemic, which is still useful for our purpose of understanding how the query topic relates to it.

This process of concept forging can be taken further through analysis of the documents retrieved by the Semantic Search Pipeline. By analyzing the terms in the retrieved documents that have the highest TF-IDF scores, it is possible to identify additional terms related to the search topic (terms perhaps not

[3]In fact, 'corona' gained an entirely new sense post-Covid, referring to the novel coronavirus disease, where previously it had referred either to a circle of diffracted light seen around the sun or moon, or to a beer company.

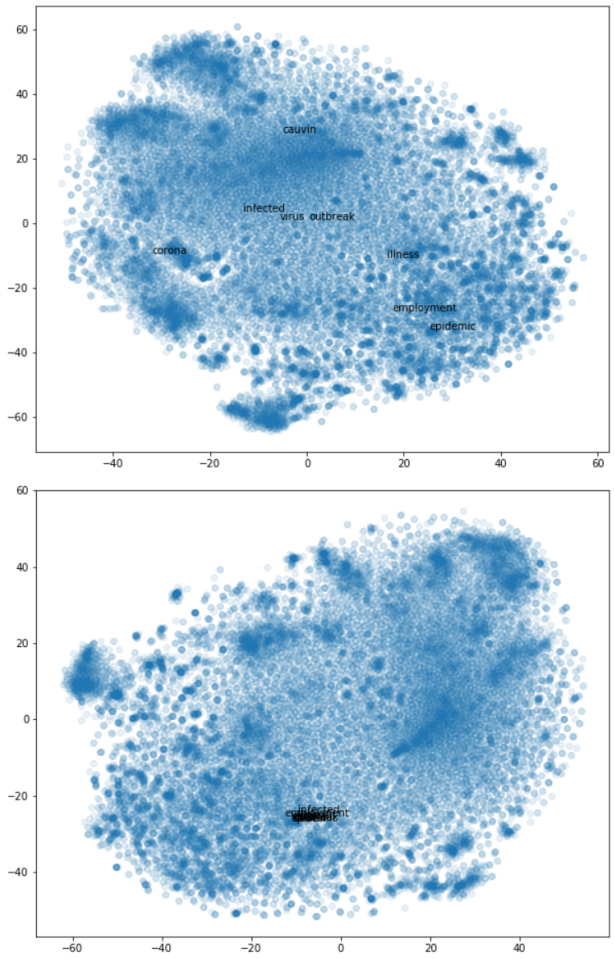

Fig. 6. Semantic vector space, 50 dimensions transformed to 2 using t-SNE; January 2020 (top) and April 2020 (bottom)

identified by the pipeline itself). These terms can be used as input to the pipeline to expand the search further, creating a sort of feedback loop to improve and expand a search until the user is confident that the results are sufficiently exhaustive. It is worth testing in future work the value of this type of analysis. An even more promising source of potential insight is using the documents retrieved by the pipeline as labeled data to build a classifier for the search topic. This is especially helpful for more complex topics, where keywords paint far from a complete picture, allowing us to see what calls mentioning Covid-19 tended to be about.

VI. Conclusion

A high-quality search improves the quality of any subsequent analysis based on its results. Expanding the query to include more keywords is helpful up to a point, but certain relevant documents will not contain any obvious keywords to associate them with the search topic, and so may never be found using keyword searches. Augmenting our pipeline with a near neighbor search allows us to find documents based on overall document similarity, while still keeping the search process computationally inexpensive through LSH.

The Semantic Search Pipeline both increased recall by 31.9% for our dataset, and improved the representativeness of the results, allowing us to get a deeper understanding of the concept of 'Covid-19' as it relates to this data. The pipeline allows our searches to react quickly to sudden changes resulting from events like a pandemic, natural disaster, or other major event. Future work includes scaling up to a larger portion of the database, further formalizing the process of *concept forging*, and application to other search topics like intelligibility problems, where the concept is expressed in multiple ways and is difficult to summarize in keywords.

References

[1] S. Kuzi, A. Shtok, and O. Kurland, "Query expansion using word embeddings," *International Conference on Information and Knowledge Management, Proceedings*, pp. 1929–1932, 2016.

[2] D. Roy, D. Paul, M. Mitra, and U. Garain, "Using Word Embeddings for Automatic Query Expansion," *Neu-IR '16 SIGIR Workshop on Neural Information Retrieval*, 2016.

[3] F. Diaz, B. Mitra, and N. Craswell, "Query Expansion with Locally-Trained Word Embeddings," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1 (Long Pa, pp. 367–377, 2016.

[4] F. Pan, "Document Retrieval in Big Data," *IMMM 2014, The Fourth International Conference on Advances in Information Mining and Management*, no. c, pp. 79–82, 2014.

[5] H. Li, W. Liu, and H. Ji, "Two-stage hashing for fast document retrieval," *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, vol. 2, pp. 495–500, 2014.

[6] J. R. Firth, "A synopsis of linguistic theory 1930–1955," in *Studies in linguistic analysis*, 1957, pp. 1–32.

[7] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *Proceedings of the 31 st International Conference on Machine Learning, Beijing, China*, 2014.

[8] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," *STOC*, pp. 604–613, 1998.

[9] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," *Proceedings of the IEEE International Conference on Computer Vision*, no. Iccv, pp. 2130–2137, 2009.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pp. 1–12, 2013.

[11] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2011.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," no. Mlm, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[13] W. L. Hamilton, J. Leskovec, and D. Jurafsky, "Diachronic word embeddings reveal statistical laws of semantic change," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, Aug. 2016, pp. 1489–1501.

[14] G. Recchia, E. Jones, P. Nulty, J. Regan, and P. de Bolla, "Tracing shifting conceptual vocabularies through time," *Lecture Notes in Computer Science*, vol. 10180 LNAI, pp. 19–28, 2017.

[15] A. Kutuzov, L. Øvrelid, T. Szymanski, and E. Velldal, "Diachronic word embeddings and semantic shifts: a survey," in *Proceedings of the 27th International Conference on Computational Linguistics*, Aug. 2018, pp. 1384–1397.

[16] A. Rosenfeld and K. Erk, "Deep neural models of semantic shift," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Jun. 2018, pp. 474–484.

[17] M. D. Tredici, R. Fernández, and G. Boleda, "Short-term meaning shift: A distributional exploration," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 2069–2075, 2019.