Evaluating retrieval performance of search algorithms

Josh Malina

Abstract

Dense, distributed representations of semantic space, or embeddings, have changed natural language processing and information retrieval. Despite these advances, traditional, sparse representations of documents in informational retrieval algorithms still remain popularly used and highly competitive. In this paper, we evaluate the performance of a small set of approaches to search. We show differences in performance across search missions, and conclude that a hybrid approach that uses both word embeddings and more traditional term frequency weights is superior in some instances.

Representing Documents

One representation of documents is the bag-of-words model, where a document is represented as the unsorted set of its words. When implemented as a sparse vector, it is normally the length of the vocabulary, and words are mapped to numbers at their indices: either binary (1 if it appears in the document, 0 otherwise), as its frequency in the document, or by Term Frequency - Inverse Document Frequency (TF-IDF).

TF-IDF refers to a family of algorithms that attempts to estimate a word's importance based on its representation in the document and the corpus as a whole. In one form, it is the frequency within the document divided by the inverse of its frequency in the corpus (i.e. how many documents it appears in at least once) and the maximum word count of the most popular word in the document. Although an old algorithm, TF-IDF has been hard to beat on search tasks.

$$tfidf(w_i, d) = \frac{tf(w_i)}{idf(w_i) * MaxWordCount(d)}$$

An update to TF-IDF is BM25, which is also a sparse-vector approach, but which includes two corpus specific constants, *b* and *k*, which can be learned ahead of time, as well as an average document length term, which seeks to correct for scores biased in the direction of longer documents:

$$bm25(w_i, d, k, b) = idf(w_i)\frac{tf(w_i)(k+1)}{tf(w_i) + k(1 - b + b(\frac{|D|}{avgdl}))}$$

where |*D*| is the length of the document, and *avgdl* is the average length of a document in the corpus ("Okapi BM25." *Wikipedia*. Wikimedia Foundation, n.d. Web. 30 July 2016.).

More recently, machine learning approaches have ushered in the widespread use of "word embeddings" to represent semantic relationships in the vocabulary (Mikolov 2013). These are real-valued, dense vector representations of words that can improve document models. According to the Skip-gram word embedding model, words that keep the same neighbors (or contexts) are pushed around in a high dimensional space with the goal of creating groups of highly synonymous words that are close to each other and relatively far (in Euclidean distance, or cosine distance) from words that could not be as easily interchanged. Famously, well trained word embeddings have been able to complete word analogy pairs, where the *vector_king - vector_man + vector_woman* returns as the nearest neighbor, *vector_queen.*

We have developed an algorithm, *Algorithm 1*, which defines a document in terms of the TF-IDF weighted sum of the word embeddings matching the words in the document.

$$D_{emb} = normalize \sum_{w}^{W} tfidf(w) * embedding(w)$$

where *w* is a word in document *D.* This leverages the semantic power of the word embedding with a tf-idf weight, to add information about how important the word is to the document.

The so-called skip-gram architecture can also create "pure" document embeddings by giving it document, word edges. Instead of training on word-word co-occurrence, the network can be fed any any bipartite graph (Perozzi 2014). We have used this "Deep Walk" approach to create embeddings for documents themselves, where the edges to the graph are of type *(documentID -> word_in_document)*. We've also implemented another document representation, inspired by the Parargraph2Vec model (Mikolov 2014). In that case, the edges include both *(documentID -> word_in_document)* and the traditional Skip-gram edges of *(word -> neighboring word)*. We've also used these two "pure" approaches to come up with an additional weighting scheme, in which the original skip gram word embeddings are weighted according to the dot product of the same deep walk word embedding with the document it appears in. This, again, is a measure of the importance of the word to the document:

$$D_{emb_{deep}} = normalize \sum_{w_{skip}}^{W} (w_{deep} \cdot d_{deep})(embedding(w_{skip}))$$

,

where w_skip represents a word embedding from the skip gram model.

$$D_{emb_{d2v}} = normalize \sum_{w_{skip}}^{W} (w_{d2v} \cdot d_{d2v})(embedding(w_{skip}))$$

,

This is same algorithm, but instead using Doc2Vec/Paragraph2Vec weights.

Algorithm Summary

Name	Kind
Tf-Idf	Sparse
BM-25	Sparse
Ketos Original	Dense Hybrid
Doc2Vec	Dense Pure
DeepWalk	Dense Pure
Doc2Vec Weighted	Dense Hybrid
DeepWalk Weighted	Dense Hybrid

Scoring

For all the myriad representations of documents, the scoring function we've chosen for ranking results of a search query is the same -- the documents' cosine similarity:

$$\frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

where *A* and *B* are document vectors.

Search Task

Performance was evaluated on an open source Reuters news corpus of 12,895 documents (Reuters21578-Apte-90Cat) which are hand labeled from a collection of 90 keywords. From this corpus, we used a subset (87%) of the documents that were labeled with a single keyword, since those documents seemed to enjoy a higher correspondence with their labels than those with multiple keywords.

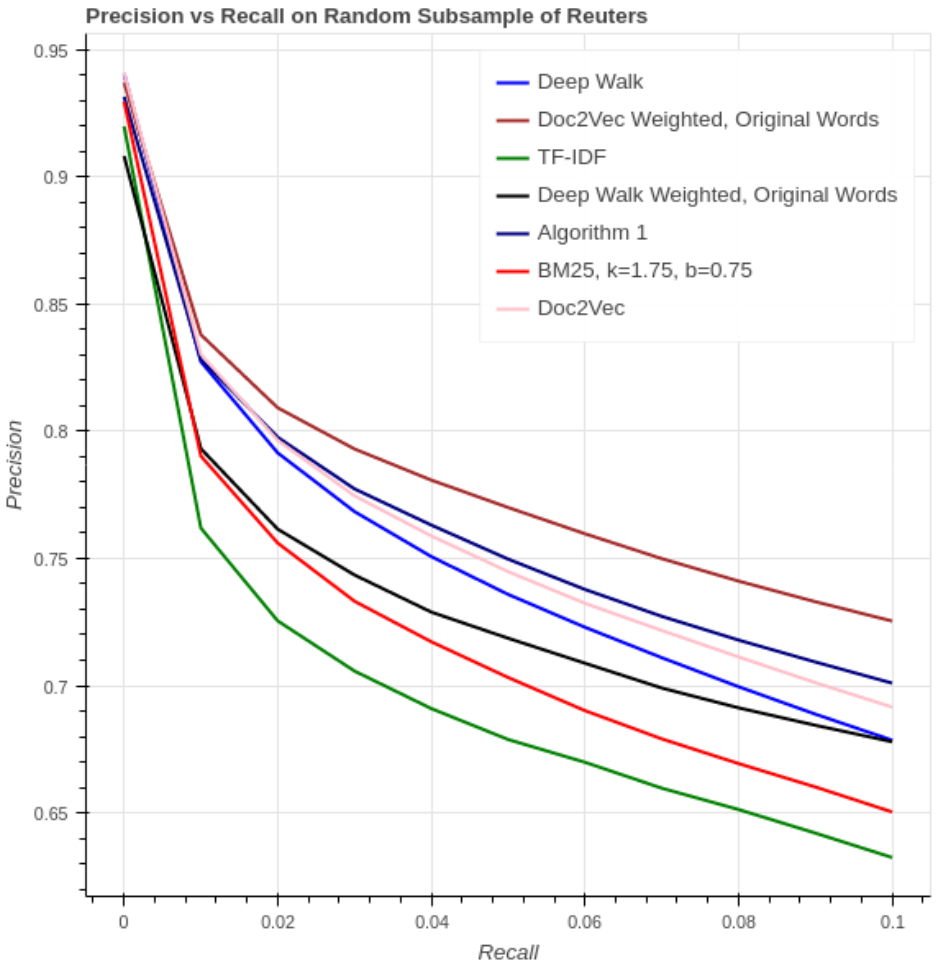
To keep evaluation times manageable, we further subsetted a random, size 1000 set of "query documents", and used the remaining ~10,000 "search documents" to be the result set. For each query document, cosine similarity scores were computed for each document in the search space, sorted according to highest affinity, and then labeled 1 or 0 depending on whether they were considered relevant. In this first search task, a document from the result set is considered relevant if it shares a keyword with the query document. Thus, if the query document has been annotated with the tag "cotton", a relevant document also shares this label.

Evaluation

From these sets of queries and their results, precision and recall scores were calculated. Precision is the proportion of results that are relevant, whereas recall is the proportion of relevant results. Since our methodology returns all documents for every query (and since ordinary users of a search engine rarely proceed past the first page or so of results), it was useful to calculate precision and recall at specific intervals rather than for entire result set. By plotting precision against recall, we can see how the performance of the algorithm varies as we get further and further from the first search result.

First Mission Results

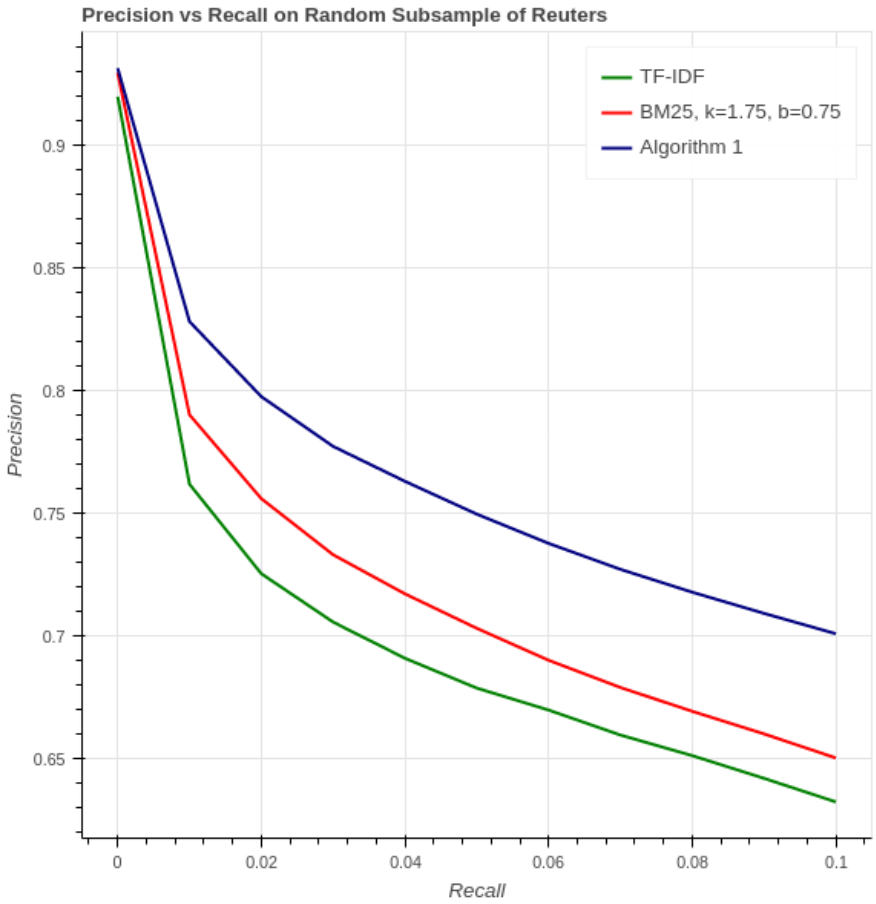
The following set of plots show the performance of the algorithms on the first search mission, as defined above. Multiple plots are shown since the groupings between lines, especially at the beginning, are quite close. As can be seen, the algorithms are quite competitive, with initial levels of precision very high and very similar.



All methods

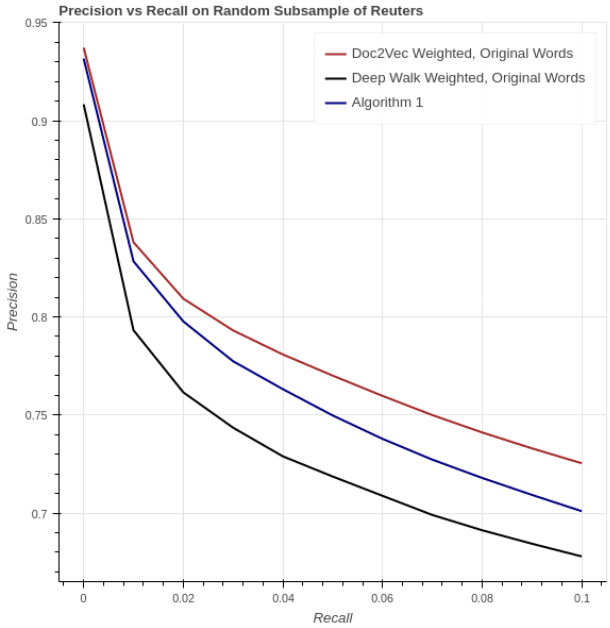
In the figure shown above, we plot an interpolated precision curve against 10 standardized recall points. The precision curve is interpolated for every query, and then averaged over all 1000 queries for each algorithm. The precision curve is calculated over a subset of the recall points, ranging from 0 to 0.1, which emphasizes the performance of the algorithms only in the first fraction of search results -- which, we assume, reflects the set of the results that most users are interested in. In this first plot, the methods that seem to produce the most relevant results the soonest are Doc2Vev and Doc2Vec Weighted, although they are closely followed by other embedding, hybrid and sparse vector based approaches. TF-IDF, however, does not seem to be performing as well.

Algorithm 1 vs Traditional Methods



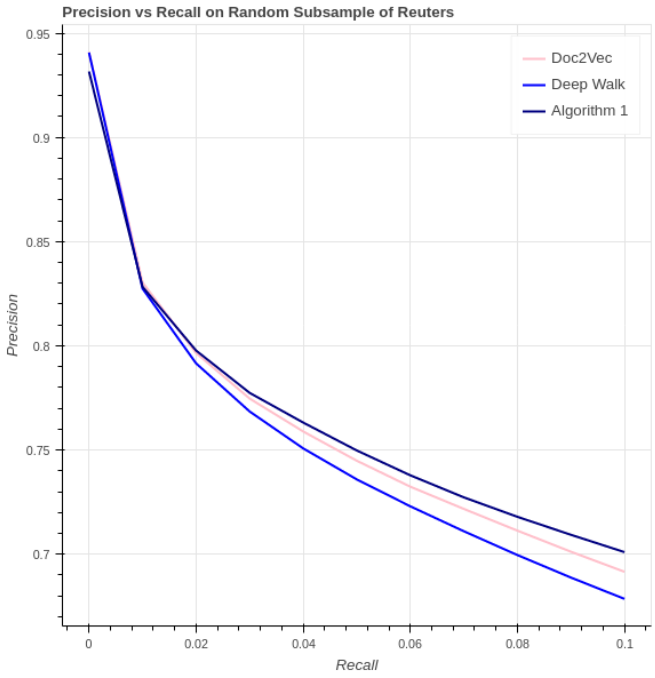
Here, we see that our method, Algorithm 1, clearly outperforms the sparse vector approaches of TF-IDF and BM25 -- although, at least initially, BM25 and Algorithm 1 are achieving similar levels of precision.

Hybrid Methods



In this plot, we can see the performance of the hybrid plots. All are Skip-gram word embeddings combined using different measures of weight. Here, Doc2Vec Weighted outperforms both our algorithm, Algorithm 1, and the very similar Deep Walk Weighted.

Pure approaches vs Algorithm 1

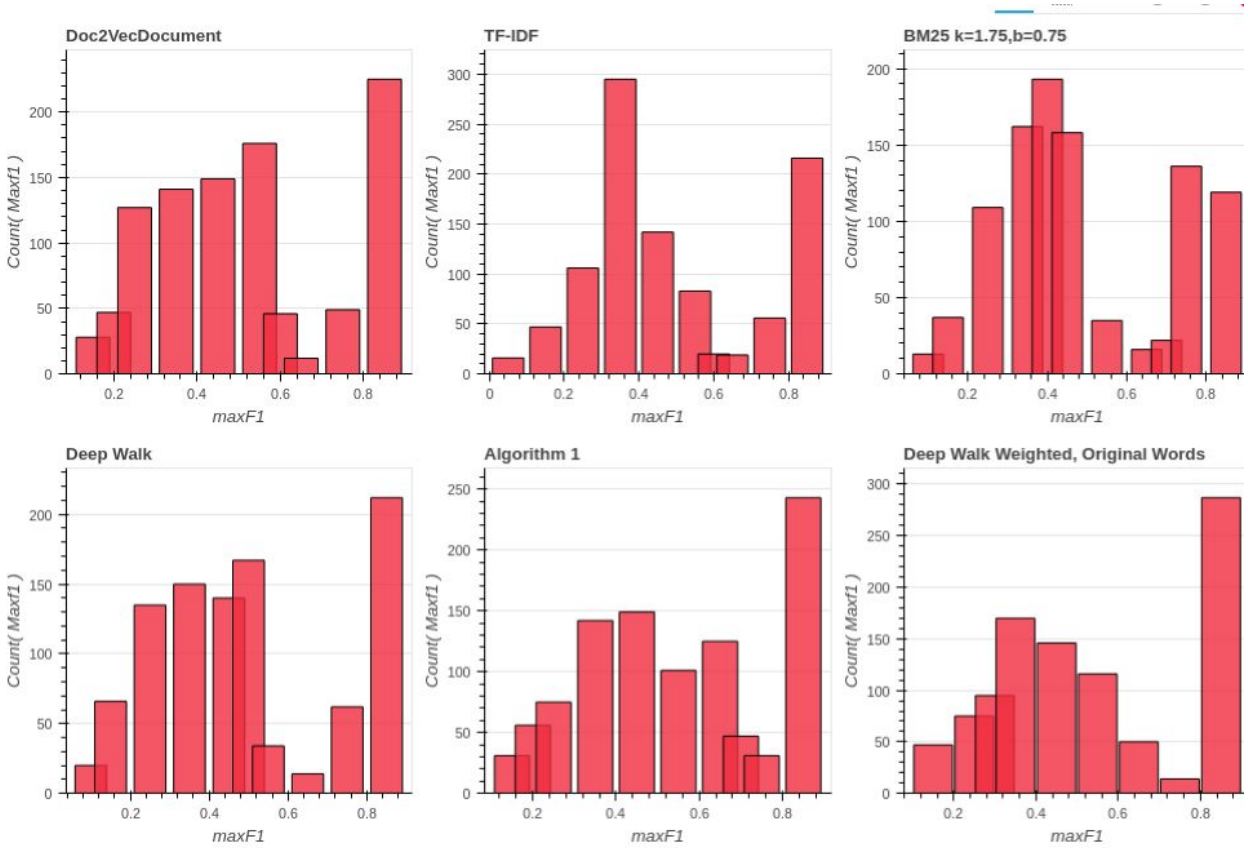


In this figure, we are comparing the “pure” approaches to our hybrid algorithm, Algorithm 1. Although the pure approaches initially outperform Algorithm 1 at the beginning of the search

results, Algorithm 1 eventually catches up and then exceeds both of the pure approaches. This suggests the possibility of ensembling methods to get the best performance overall.

Testing Search Mission 1

Although it is hard to summarize the performance of these models by a single number, it is informative to try and clarify the differences between the performance curves with statistical hypothesis testing.



The distribution of max F1 scores across 6 of the 7 algorithms. The last distribution (not shown) has a similar shape. The F1 score is the harmonic mean of precision and recall, and tracks both high and precision and recall curves when it itself is high, so it serves as a good candidate for testing. The max F1 score for a given query is the highest F1 score obtained for any query for a given method. Although the sample size is large (1000), the distributions are not normal -- in fact, they are bimodal, which suggests the use of a nonparametric test. Note that the distributions represent MaxF1 scores over the entire curve, whereas the line graphs represent a truncated view of the search space, namely, from recall points 0.0 to 0.1.

```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 89.3063, df = 6, p-value = 0
```

Comparison of x by group
(Bonferroni)

Col Mean- Row Mean	Algorith	BM25	D2V	Deep Wal	DeepWWei	Doc2VecW
BM25	5.125541 0.0000					
D2V	0.493783 1.0000	-4.631757 0.0000				
Deep Wal	1.341686 1.0000	-3.783854 0.0016	0.847903 1.0000			
DeepWWei	1.042387 1.0000	-4.083153 0.0005	0.548604 1.0000	-0.299299 1.0000		
Doc2VecW	-2.019708 0.4558	-7.145250 0.0000	-2.513492 0.1255	-3.361395 0.0081	-3.062096 0.0231	
TF-IDF	5.468998 0.0000	0.343457 1.0000	4.975215 0.0000	4.127312 0.0004	4.426611 0.0001	7.488707 0.0000

Given, the non-normality of the methods' max-F1 distributions, we conducted a one-way non parametric test (Kruskal-Wallis) to evaluate a group-wide difference in distributions. Following the rejection of the null here, we completed a group of pair-wise Dunn tests with multiple comparison correction. We can see from the results that many of the pairs of algorithms, including (Dov2Vec, Algorithm1) and (DeepWalk, Algorithm1) are not statistically significantly different, despite varying in the plots.

However, we are able to observe a statistically significant difference between Algorithm 1 and the two sparse vector based approaches, TF-IDF, and BM25, so we can be justified that our hybrid approach performs better than those traditional approaches. It is not clear, however, which of the hybrid or pure approaches performs the best.

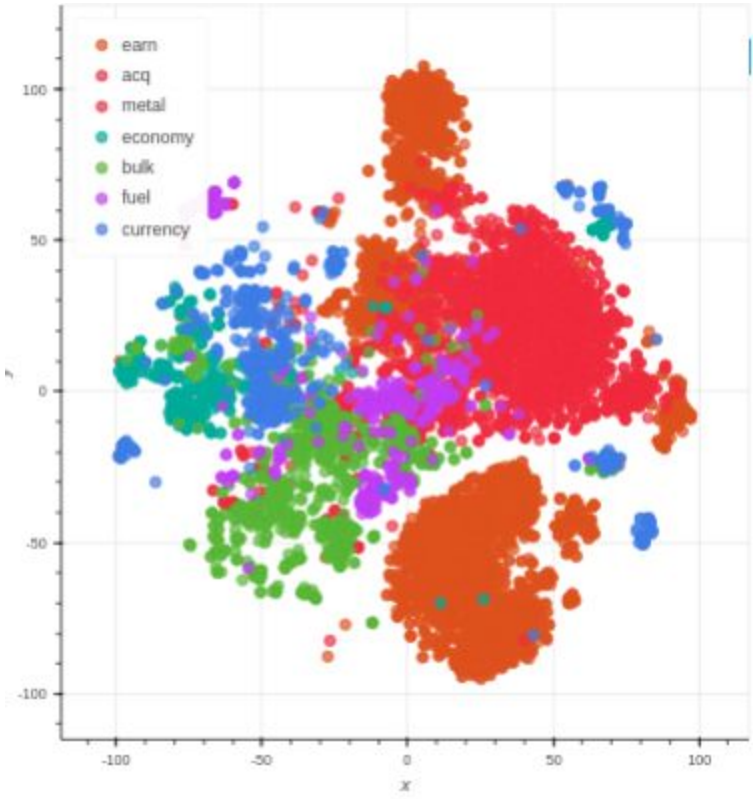
Second Search Mission

A secondary search task was also performed. In this case, the 90 categories were aggregated into 7 umbrella groups, and a document from the result set was considered relevant if it shared any keyword tags that belonged to the same category as the query tag.

For example, if a query was labeled with "rice", which belonged to the "bulk" category, any search document that was labeled "rice", "grain" or any of the other tags in the "bulk"

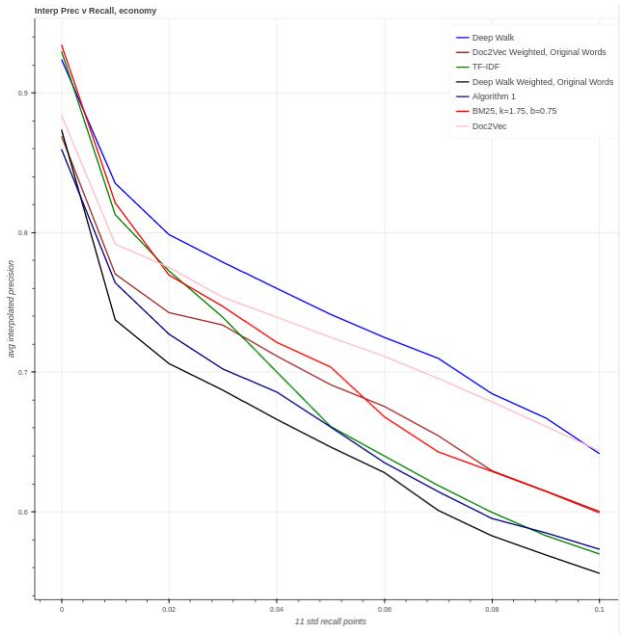
category would also be considered relevant. This “fuzzy” task was aimed at more closely resembling the way a user might define relevance -- which would include not merely a string match, as in the case that they are googling their name to see how famous they are -- but also in cases where concepts are being searched for. For example, if a query mentions only “espionage” it would still be considered appropriate to return a document only mentioning “spying”.

T-sne plot of documents by category



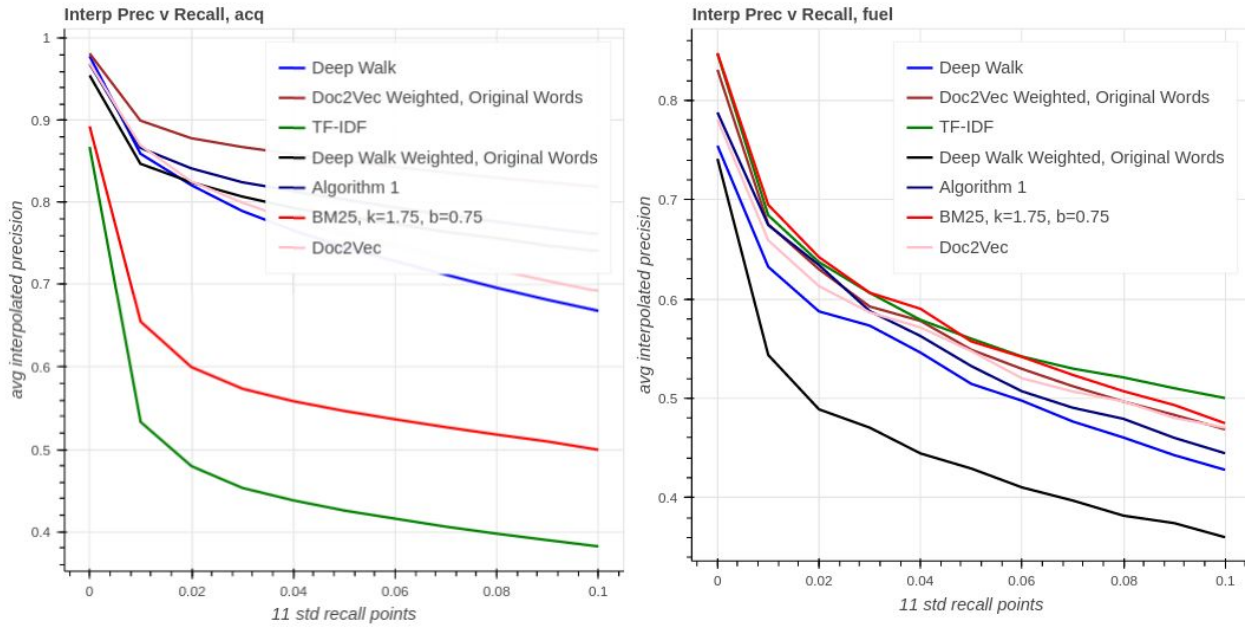
In this figure, we see clusters of articles labeled by their aggregated categories. The plot was produced by running a dimensionality reduction algorithm, t-sne, over the 100 dimensional document embeddings from Algorithm 1.

All methods on “economy” group



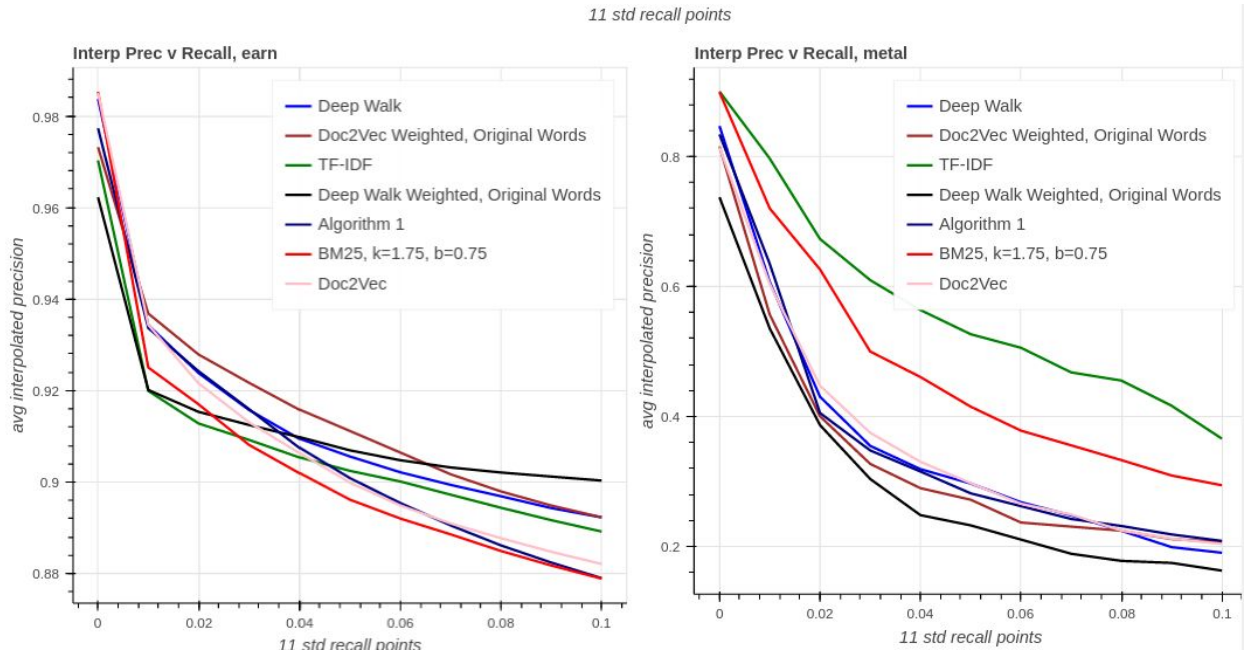
On the aggregated level, the results are less clear cut. In this plot of “economy” query results, the leading performers, initially, are Deep Walk, BM25, and TF-IDF, though they are met later by Doc2Vec.

“Acq” and “Fuel”



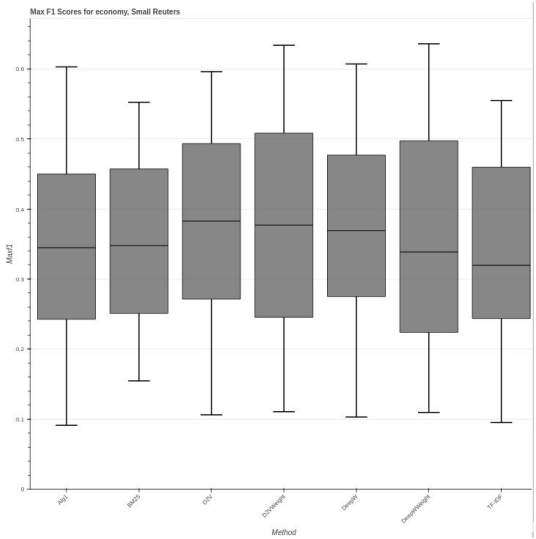
In plots of keyword tags “acq”, which stands for acquisitions, the hybrid and embedding based approaches perform much better than the traditional sparse vector approaches of BM25 and TF-IDF. In fuel, on the other hand, BM25 and TF-IDF are performing quite well.

Earn and Metal



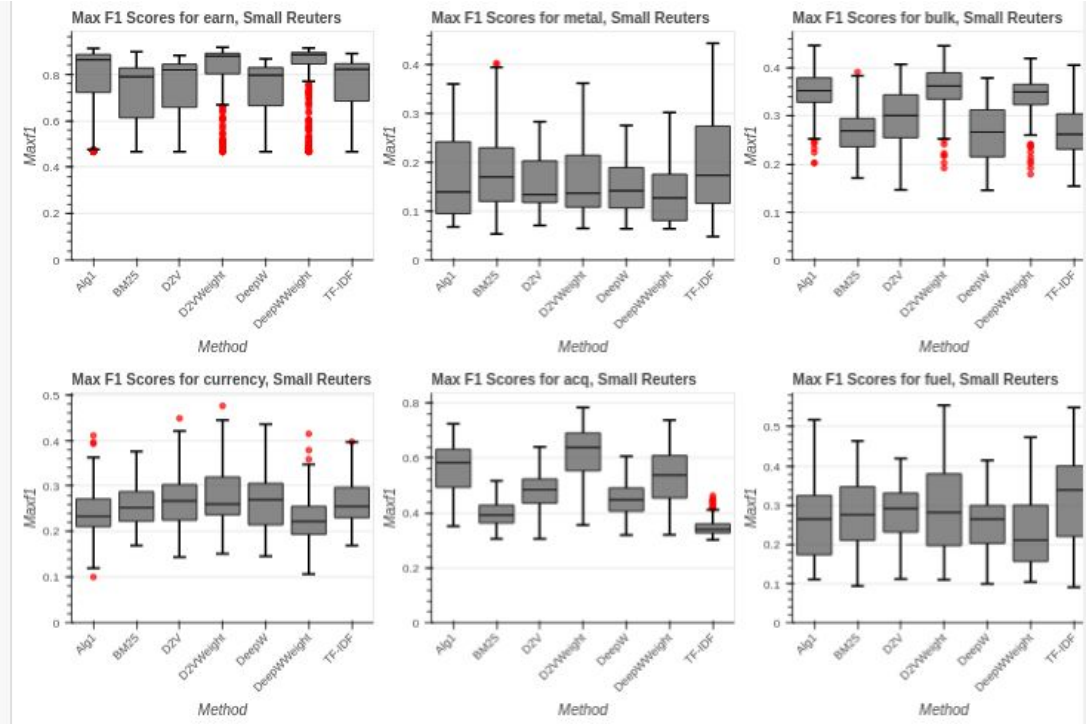
In the categories of earn and metal, we see more diverse behavior. In metal, it is clear that the traditional approaches are far superior to the more modern ones, whereas in “earn” only BM-25 can compete with the embedding based approaches.

Although slight differences exist in the performance of these algorithms, now separated by group, we might be still more confident in assessing differences after conducting some hypothesis tests. Viewing the distributions as side-by-side boxplots, we can examine performance and distribution.



This figure shows box plots for the distributions of the max F1 scores for economy. The distributions are not so distinct from each other, though there does appear to be some methods that perform better than others. However, in examining the hypothesis test, this no longer seems clear. A kruskal wallis test could not reject the null, with a p-value of 0.68

.



In this figure, we see the remaining categories and the distribution of their MaxF1 scores. Of the seven categories, we could reject the null hypothesis that all seven algorithms performed the same on five of them.

Aggregated Category	**Kruskal wallis p-value**	**Family Wise**	**Reject**
economy	0.6843	0.00714285714	no
bulk	< 2.2e-16	0.00714285714	yes ***
acq	< 2.2e-16	0.00714285714	yes ***
earn	< 2.2e-16	0.00714285714	yes ***
currency	0.0007452	0.00714285714	yes **
metal	0.1653	0.00714285714	no
fuel	0.005307	0.00714285714	yes *

This figure summarizes the conclusions of the one-way tests on the categories. The error rate was adjusted by Bonferroni correction to accommodate the 7 tests. Although most categories were either clearly significant or not, the category “fuel” could just barely be judged significant.

Conclusions

In overall search results, we found that embedding and hybrid based approaches outperform more traditional TF-IDF and BM-25 approaches in a statistically significant way some of the time. However, when breaking down the search queries into aggregate categories, the results were less clear. Sometimes, the hybrid/modern approaches were clearly superior, while at other times, the traditional approaches were far in a way much better.

Although we were able to state statistically significant conclusions about the differences between some of the algorithms some of the time, it is hard to explain why their success was contextual. It could be because of **the way the algorithms work**: embedding based approaches return fuzzy matches, even if that is not always the intention of the search user. Some query documents may have, as an artifact of the data source, more or less fuzzy matches. For example, the various synonyms of the word “earn”, or other nearest neighbors of financial terminology may have factored prominently in the success of the embedding approaches in this case.

There is also a problem with the way **relevance is defined.** In our corpus, we have left that to human annotators of articles. But a user might define relevance differently. Some keywords may have been more exhaustively used to tag articles than others.

In the end, to be more confident, we need to evaluate these algorithms on more data sets and on more search missions. There exist other data sources that more stringently define relevance based on a set of criteria.

We should also segment our analysis into categories that are specific to user groups, corresponding to different sections of the precision / recall curve. Ordinary users may find early, high precision results acceptable while another user, such as a lawyer, where all relevant results in a corpus of legal documents might affect preparation for a case, would like an algorithm that is optimized for high recall.

Finally, if we find evidence for significant performance differences that are data-dependent, we can try to ensemble some of the models for more performance gains.

References

Le, Quoc and Mikolov, Tomas. *Distributed Representations of Sentences and Documents Proceedings of the 31 st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.*

Mikolov, T., Chen, K., Corrado, G. and Dean, J. *Efficient estimation of word representations in vector space* arXiv:1301.3781, 2013

"Okapi BM25." *Wikipedia*. Wikimedia Foundation, n.d. Web. 30 July 2016.

Perozzi, B., Al-Rfou, R., and Skiena, Steven. *DeepWalk: Online Learning of Social Representations,* arXiv:1403.6652, 2014

R Output
HUID: 20921773

Running pairwise non-parametric tests over maxF1 distributions of the algorithms

```
df = read.csv(file.choose())
View(df)
dunn.test(df$maxF1T, df$method,alpha=0.05,wrap=TRUE,method="bonferroni")$comparisons

 Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 89.3063, df = 6, p-value = 0


                      Comparison of x by group
                           (Bonferroni)
Col Mean-|
Row Mean |   Algorith       BM25        D2V   Deep Wal   DeepWWei   Doc2VecW
---------+--------------------------------------------------------------------
   BM25 |   5.125541
        |     0.0000
        |
    D2V |   0.493783  -4.631757
        |     1.0000     0.0000
        |
Deep Wal |   1.341686  -3.783854   0.847903
        |     1.0000     0.0016     1.0000
        |
DeepWWei |   1.042387  -4.083153   0.548604  -0.299299
        |     1.0000     0.0005     1.0000     1.0000
        |
Doc2VecW |  -2.019708  -7.145250  -2.513492  -3.361395  -3.062096
        |     0.4558     0.0000     0.1255     0.0081     0.0231
        |
  TF-IDF |   5.468998   0.343457   4.975215   4.127312   4.426611   7.488707
        |     0.0000     1.0000     0.0000     0.0004     0.0001     0.0000

 [1] "Algorithm1 - BM25"           "Algorithm1 - D2V"            "BM25 - D2V"                  "Algorithm1 - Deep Walk"
 [5] "BM25 - Deep Walk"            "D2V - Deep Walk"             "Algorithm1 - DeepWWeight"    "BM25 - DeepWWeight"
 [9] "D2V - DeepWWeight"           "Deep Walk - DeepWWeight"     "Algorithm1 - Doc2VecWeight"  "BM25 - Doc2VecWeight"
[13] "D2V - Doc2VecWeight"         "Deep Walk - Doc2VecWeight"   "DeepWWeight - Doc2VecWeight" "Algorithm1 - TF-IDF"
[17] "BM25 - TF-IDF"               "D2V - TF-IDF"                "Deep Walk - TF-IDF"          "DeepWWeight - TF-IDF"
[21] "Doc2VecWeight - TF-IDF"


boxByCat <- function(cat) {
  boxplot(df3[df3$aggCat==cat,]$maxF1 ~ df3[df3$aggCat==cat,]$method, main=paste("MaxF1 scores for ",cat), ylab="MaxF1")
}


              Kruskal-Wallis rank sum test

data:  df3[df3$aggCat == "economy", ]$maxF1 and df3[df3$aggCat == "economy", ]$method
Kruskal-Wallis chi-squared = 3.9438, df = 6, p-value = 0.6843



> test_by_cat("bulk")
```

```
        Kruskal-Wallis rank sum test

data:  df3[df3$aggCat == cat, ]$maxF1 and df3[df3$aggCat == cat, ]$method
Kruskal-Wallis chi-squared = 163.09, df = 6, p-value < 2.2e-16


> test_by_cat("metal")

        Kruskal-Wallis rank sum test

data:  df3[df3$aggCat == cat, ]$maxF1 and df3[df3$aggCat == cat, ]$method
Kruskal-Wallis chi-squared = 9.1508, df = 6, p-value = 0.1653


> test_by_cat("earn")

        Kruskal-Wallis rank sum test

data:  df3[df3$aggCat == cat, ]$maxF1 and df3[df3$aggCat == cat, ]$method
Kruskal-Wallis chi-squared = 557.62, df = 6, p-value < 2.2e-16

> test_by_cat("currency")

        Kruskal-Wallis rank sum test

data:  df3[df3$aggCat == cat, ]$maxF1 and df3[df3$aggCat == cat, ]$method
Kruskal-Wallis chi-squared = 23.158, df = 6, p-value = 0.0007452

acq
        Kruskal-Wallis rank sum test

data:  df3[df3$aggCat == cat, ]$maxF1 and df3[df3$aggCat == cat, ]$method
Kruskal-Wallis chi-squared = 827.33, df = 6, p-value < 2.2e-16


> test_by_cat("fuel")

        Kruskal-Wallis rank sum test

data:  df3[df3$aggCat == cat, ]$maxF1 and df3[df3$aggCat == cat, ]$method
Kruskal-Wallis chi-squared = 18.4, df = 6, p-value = 0.005307


---

dunnByCat <- function(cat) {
  dunn.test(df3[df3$aggCat==cat,]$maxF1, df3[df3$aggCat==cat,]$method,alpha=0.007,wrap=TRUE,method="bonferroni")
}


> dunnByCat("bulk")
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 163.0881, df = 6, p-value = 0


                        Comparison of x by group
                            (Bonferroni)
Col Mean-|
```

```
Row Mean |      Alg1       BM25        D2V   D2VWeigh      DeepW   DeepWWei
---------+-----------------------------------------------------------------
    BM25 |   7.237061
         |     0.0000
         |
     D2V |   4.539137  -2.697924
         |     0.0001     0.0733
         |
D2VWeigh |  -0.692486  -7.929548  -5.231624
         |     1.0000     0.0000     0.0000
         |
   DeepW |   7.400445   0.163383   2.861307   8.092931
         |     0.0000     1.0000     0.0443     0.0000
         |
DeepWWei |   0.724434  -6.512627  -3.814702   1.416921  -6.676010
         |     1.0000     0.0000     0.0014     1.0000     0.0000
         |
  TF-IDF |   7.284423   0.047361   2.745285   7.976910  -0.116021   6.559988
         |     0.0000     1.0000     0.0635     0.0000     1.0000     0.0000

> dunnByCat("acq")
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 827.331, df = 6, p-value = 0


                     Comparison of x by group
                           (Bonferroni)
Col Mean-|
Row Mean |      Alg1       BM25        D2V   D2VWeigh      DeepW   DeepWWei
---------+-----------------------------------------------------------------
    BM25 |   14.70089
         |     0.0000
         |
     D2V |   6.345791  -8.355099
         |     0.0000     0.0000
         |
D2VWeigh |  -3.643321  -18.34421  -9.989112
         |     0.0028     0.0000     0.0000
         |
   DeepW |   9.343118  -5.357773   2.997326   12.98643
         |     0.0000     0.0000     0.0286     0.0000
         |
DeepWWei |   2.385687  -12.31520  -3.960104   6.029008  -6.957430
         |     0.1790     0.0000     0.0008     0.0000     0.0000
         |
  TF-IDF |   19.84023   5.139339   13.49443   23.48355   10.49711   17.45454
         |     0.0000     0.0000     0.0000     0.0000     0.0000     0.0000

> dunnByCat("earn")
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 557.62, df = 6, p-value = 0


                     Comparison of x by group
                           (Bonferroni)
Col Mean-|
Row Mean |      Alg1       BM25        D2V   D2VWeigh      DeepW   DeepWWei
```

```
---------+--------------------------------------------------------------------
   BM25 |   10.92433
        |    0.0000
        |
    D2V |   8.025795  -2.898543
        |    0.0000    0.0394
        |
D2VWeigh |  -3.536444  -14.46078  -11.56224
        |    0.0043    0.0000    0.0000
        |
  DeepW |   10.43109  -0.493247   2.405296   13.96753
        |    0.0000    1.0000    0.1697    0.0000
        |
DeepWWei |  -5.318523  -16.24286  -13.34431  -1.782078  -15.74961
        |    0.0000    0.0000    0.0000    0.7847    0.0000
        |
 TF-IDF |   7.697547  -3.226791  -0.328247   11.23399  -2.733543   13.01607
        |    0.0000    0.0131    1.0000    0.0000    0.0658    0.0000

> dunnByCat("currency")
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 23.1581, df = 6, p-value = 0


                      Comparison of x by group
                           (Bonferroni)
Col Mean-|
Row Mean |      Alg1       BM25        D2V   D2VWeigh      DeepW   DeepWWei
---------+--------------------------------------------------------------------
   BM25 |  -1.212724
        |    1.0000
        |
    D2V |  -2.068538  -0.855814
        |    0.4052    1.0000
        |
D2VWeigh |  -2.325667  -1.112942  -0.257128
        |    0.2104    1.0000    1.0000
        |
  DeepW |  -1.903516  -0.690792   0.165022   0.422150
        |    0.5982    1.0000    1.0000    1.0000
        |
DeepWWei |   1.547566   2.760290   3.616105   3.873233   3.451082
        |    1.0000    0.0606    0.0031    0.0011    0.0059
        |
 TF-IDF |  -1.612807  -0.400083   0.455730   0.712859   0.290708  -3.160374
        |    1.0000    1.0000    1.0000    1.0000    1.0000    0.0165

> dunnByCat("fuel")
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 18.3963, df = 6, p-value = 0.01


                      Comparison of x by group
                           (Bonferroni)
Col Mean-|
Row Mean |      Alg1       BM25        D2V   D2VWeigh      DeepW   DeepWWei
---------+--------------------------------------------------------------------
```

```
    BM25 |  -0.617799
         |     1.0000
         |
     D2V |  -0.719715  -0.101915
         |     1.0000     1.0000
         |
D2VWeigh |  -0.931427  -0.313627  -0.211712
         |     1.0000     1.0000     1.0000
         |
   DeepW |   0.646168   1.263968   1.365883   1.577595
         |     1.0000     1.0000     1.0000     1.0000
         |
DeepWWei |   1.371137   1.988937   2.090853   2.302565   0.724969
         |     1.0000     0.4904     0.3837     0.2237     1.0000
         |
  TF-IDF |  -2.491687  -1.873887  -1.771971  -1.560259  -3.137855  -3.862824
         |     0.1335     0.6399     0.8022     1.0000     0.0179     0.0012
```