Text Retrieval by Term Co-occurrences in a Query-based Vector Space

Eriks Sneiders
Department of Computer and Systems Sciences
Stockholm University
Postbox 7003, SE-164 07, Kista, Sweden
eriks@dsv.su.se

Abstract

Term co-occurrence in a sentence or paragraph is a powerful and often overlooked feature for text matching in document retrieval. In our experiments with matching email-style query messages to webpages, such term co-occurrence helped greatly to filter and rank documents, compared to matching document-size bags-of-words. The paper presents the results of the experiments as well as a text-matching model where the query shapes the vector space, a document is modelled by two or three vectors in this vector space, and the query-document similarity score depends on the length of the vectors and the relationships between them.

1 Introduction

Vector-space in text retrieval is old news. Cosine similarity by Salton and McGill (1986), along with the probabilistic retrieval model by Robertson and Spark Jones (1976), has dominated text retrieval for the last three decades. Much development has focused on improving the relevance scoring factors beyond term frequency and inverted document frequency, e.g., co-occurrence and proximity of query terms in the document help improve the relevance score (see Section 6).

This paper reports the results of an inductive research. We started with a practical task, then analyzed the prototype and spotted a text retrieval model, then twisted the model in order to improve it. The practical task was to increase automation of the “ask us” function on a municipal website: before the user submits his or her message, the system checks whether some answer-relevant pages are published on the website; if yes, the system delivers the pages to the user. Reuse of previously published information would save the user’s time and the municipality’s resources.

The municipality gave us 25 anonymized sample messages to experiment with, which was not enough for machine learning and text categorization methods. We knew that text-pattern matching yields superior results in automated email answering (Sneiders, 2016). Unfortunately, text patterns require learning, good text patterns are difficult to develop and maintain. We needed something new.

Our research problem is matching email-style query messages to webpages by a technique that does not require learning or development of a knowledge base. Our solution is a text-matching technique that relies on term co-occurrence in a limited text chunk (a sentence or a paragraph), yet without the complexity of good text patterns. The novelty and contribution of this paper are: (i) demonstration of the advantage of such term co-occurrence in text matching over matching of document-size bags-of-words, and (ii) a text-matching model where the query shapes the vector space, a document is modelled by two or three vectors in this vector space, and the query-document similarity score depends on the length of the vectors and the relationships between them. The model is easy to visualize.

Further in this paper, the next section presents the initial prototype, the test data, and the results of the initial tests. Sections 3 and 4 explain the text-matching model and show new test results. Section 5 discusses the advantages and limitations of the technique. Section 6 presents related and further research. Section 7 concludes the paper.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2356–2365, Osaka, Japan, December 11-17 2016.

2 Initial Experiment

While web search is a well-established industry, matching email-style messages to web documents is a novel task. The main difference between the two tasks lies in the semantic strength of the terms in the query: while search queries consist of consciously selected keywords, a message is a short story without any indication which words embody the essence of the story.

2.1 Text Matching Method

The query – an email-style message – is a bag of words. The document is a webpage split into text chunks, where a chunk is a sentence or a block of text inside HTML block-level elements (Mozilla, 2016). Each text chunk is a bag of words. Thus, our "document collection" is a collection of text chunks. We do keep track of which chunk belongs to which document. Calculation of the query-document similarity score takes five steps; these steps were experimentally developed and refined.

Step 1. The system matches the query to each chunk separately and discovers term pairs – two terms that co-occur in the chunk and co-occur in the query. From now on, only these terms pairs are significant, single terms are ignored. We do not consider term triples, quadruples, etc. A triple is two term pairs.

Step 2. Each term t_i that belongs to a term pair T in a chunk *ch* obtains a term weight by formula (1). The term weight includes term frequency in the chunk, terms frequency in the query, inverted chunk frequency of the term, which is a counterpart of inverted document frequency in our chunk collection, as well as inverted chunk frequency of the term pair. The latter means that a term may have different weights in one chunk but in different term pairs.

$$w(t_i, T, ch) = tf(t_i, ch) \cdot tf(t_i, q) \cdot icf(t_i) \cdot icf(T) \qquad (1)$$

We experimented with the document structure attribute – title, the main text, or H1-H3 heading – for term weights and discovered that term pairs appear in these structure elements somewhat equally for relevant and non-relevant documents. Therefore we do not use the document structure attribute.

Step 3. The weight of a term t_i in a term pair T for the entire document *d* is calculated by formula (2) as a sum across the chunks in the document where the term pair exists.

$$w(t_i, T, d) = \sum_{\{ch \in d | T \in ch\}} w(t_i, T, ch) \qquad (2)$$

Step 4. The weight of a term pair T*(t_i, t_j)* in the document *d* is the sum of the weights of both terms t_i and t_j calculated by formula (3).

$$w(T(t_i, t_j), d) = w(t_i, T, d) + w(t_j, T, d) \qquad (3)$$

Step 5. The query-document similarity score is the sum of all term-pair weights of this document calculated by formula (4). The document score is largely a sum of selected term weights, but we did the summation following certain logic.

$$score_0(d) = \sum_{\{T \in d\}} w(T, d) \qquad (4)$$

Alternative steps 4 and 5. We may want to skip weighing term pairs and continue with the weights of individual terms from the term pairs. Formula (5) calculates the weight of a term t_i in the entire document *d* across all the term pairs T where t_i exists. The query-document similarity score is the sum of the weights of the unique terms from all the term pairs in the document, calculated by formula (6). N is the number of these unique terms. While formulas (4) and (6) are equivalent, formulas (3) and (4) are easier to understand whereas formulas (5) and (6) will be used further in this paper.

$$w(t_i, d) = \sum_{\{T \in d | t_i \in T\}} w(t_i, T, d) = tf(t_i, q) \cdot icf(t_i) \cdot \sum_{\{T \in d | t_i \in T\}} \left(icf(T) \cdot \sum_{\{ch \in d | T \in ch\}} tf(t_i, ch) \right) \qquad (5)$$

$$score_0(d) = \sum_{i=1}^{N} w(t_i, d) \qquad (6)$$

2.2 Experiment Data

The source of the document collection was a municipal website in Swedish. The web pages were generated by a content management system and had quite uniform appearance. With a little bit of experimenting with learned to remove the headers, footers, side bars. After removing duplicate content pages, we obtained 2607 documents. These documents were split into chunks. If a chunk was a block of text, the collection of chunks got 26 064 chunks. If a chunk was a sentence, the collection got 37 420 chunks. We applied compound splitting (Sjöbergh and Kann, 2004) and lemmatization (Carlberger and Kann, 1999) to the words in a chunk before the chunk was turned into a bag of words.

We tested two options of stop-word removal. One option was using a standard list of Swedish stop-words (Text Tools, 2016). The other option was part-of-speech (POS) filtering. Analysis of 30 most frequent lemmas that matched text patterns in automated email answering (Sneiders at al., 2014) showed that only nouns and verbs were domain-specific words. Barr et al. (2008) explored Yahoo! search queries and found that 40.2% of the terms were proper nouns, 30.9% nouns, 7.1% adjectives, 2.4% verbs. Inspired by these findings, we grouped the terms in our document collection by their POS-tags and subjectively examined which parts-of-speech felt like good candidates for descriptive term pairs. Eventually, we decided that only nouns, verbs, proper nouns, adjectives, and unrecognized terms should be left in the documents after POS filtering.

The average size and standard deviation of the block-chunks were 8.9±8.45 unique lemmatized terms after removing standard stop-words, and 8.29±7.71 terms after POS filtering; both after compound splitting. For sentence-chunks, the respective sizes were 6.44±4.21 and 6.01±3.93.

We had 25 email-style messages in Swedish as test queries. Each message was spell-checked and pre-processed as a chunk. The average size of the messages was 12.68±6.25 unique lemmatized terms after removing standard stop-words, and 10.76±4.90 terms after POS filtering. Following are three sample messages – one short, one long, and one medium long. We Google-translated the Swedish originals, with minor manual corrections, and removed location names from the text.

- “Hello, I'm looking for a summer job as a gardener in `<town>` but I do not know which homepages I can go in and look for summer jobs. Which homepages I can go to and check?”
- “Hey, we're moving to `<town>` municipality in July. Today we live in `<town>` and have our children at `<name>` school. The new house is closest to `<name>` school but this is another municipality. When I spoke to the school they told me that our children were welcome there and they now have a number of students who attend the school but live across municipal borders. It requires, however, an approval of `<town>` municipality. My question now is how it works, if we want our children to go to `<name>` school because it is closest, and that several of the children's friends go to that school too. Thanks in advance.”
- “Hello! On the pedestrian and bicycle path between `<address>` in `<town>` and `<village>` is a lamppost partially bent since a storm last fall. The post is standing but the light is directed in a wrong direction. Who is responsible for this to be corrected? Thanks for a quick response. If the lamppost number is needed I can find it out.”

We used the website’s search engine to find documents relevant to the test messages. We devised a number of search queries per test message and manually examined all the retrieved documents. We found two kinds of documents. *Closely relevant documents* could answer the message. *Related documents* had “good to know” information but did not answer the message. 3 of the test messages had no relevant information on the website. 4 messages had only closely relevant documents. 4 messages had only related documents. 14 messages had both closely relevant and related documents. Of those messages that had closely relevant documents, 8 messages had one such document and 10 had two such documents. Of those messages that had related documents, 15 messages had one to three such documents, 2 had six such documents, and 1 had eleven such documents.

2.3 Performance Measures

The system matches term pairs in the query and the document, therefore it inevitably removes relevant documents that do not have matching term pairs. Hence, we measured *recall* as the share of relevant documents that did have matching term pairs and therefore were included in the results. The average recall was calculated across the queries that had corresponding documents in the collection.

In order to quantify exposure of relevant documents to the user, we measured the *average rank* (the position in the result list) of relevant documents. We calculated the average rank for each query with non-zero recall, and then the average of these averages.

In order to simulate automated email answering, we measured *P@1* (precision among top n documents where n is 1) for closely relevant documents. The average P@1 was calculated across the queries with non-zero recall.

Our baseline method was *cosine similarity* between unigram-based document vectors. We tested cosine similarity in two settings. At first, we applied cosine similarity to all the 2607 documents and ranked them. The second option was to filter the 2607 documents by the term pairs first, which resulted in a much smaller set of documents with a higher density of relevant documents. Then we applied cosine similarity in this smaller set of documents. For cosine similarity, term frequency was counted in the entire document; inverted document frequency was calculated in the document collection. We realize that cosine similarity is a rather basic baseline. It did, however, serve the purpose. We leave a more thorough comparison of text-matching methods to further research.

Recall, the average rank of relevant documents, and cosine similarity were measured separately for closely relevant and related documents.

2.4 Test Results

The tests have three parameters – (i) two options of stop-word removal, (ii) two scopes of a text chunk, and (iii) two options of the minimum number of term pairs that the entire document (not one chunk) and the query must have in common for the document to be considered for relevance ranking. Table 1 shows the initial results according to these parameters. The average values have their standard deviation, except for P@1 where individual values are 0 or 1.

Recall and the number of selected documents. The average number of selected documents lies between 25 and 109 of the total 2607, i.e., on average the system selected 1-4% of the documents. Meanwhile, the average recall does not drop below 0.61 for closely relevant documents. The best average recall is 0.83, i.e., the selected 74 (average) of the 2607 documents contain 83% (average) of all the closely relevant documents. Recall for the related documents is lower. Because the semantic distance between the query and a related document is longer than that between the query and a closely relevant document, apparently there are fewer co-occurring terms and fewer selected messages.

Stop-words	**Avg num of selected docs**	**Avg recall**		**Avg P@1**	**Avg rank**	
		Close	**Related**		**Close**	**Related**
Documents selected by chunk-sentence, min 2 common term pairs. Ranked by score$_0$						
POS filter	41.44±98.31	0.75±0.34	0.56±0.40	0.50	3.94±4.96	4.54±4.83
Standard	**74.28±204.48**	**0.83±0.33**	**0.58±0.38**	**0.56**	**3.47±3.87**	**6.51±6.84**
Documents selected by chunk-block, min 2 common term pairs. Ranked by score$_0$						
POS filter	63.36±139.49	0.75±0.34	0.57±0.39	0.50	4.66±6.04	8.50±10.17
Standard	108.52±264.91	0.83±0.33	0.61±0.36	0.50	4.19±4.52	11.64±16.63
Documents selected by chunk-sentence, min 3 common term pairs. Ranked by score$_0$						
POS filter	25.16±64.2	0.61±0.36	0.32±0.41	0.53	2.73±3.47	4.83±5.56
Standard	47.84±147.54	0.69±0.34	0.31±0.41	0.56	2.16±1.50	5.22±5.91
Documents selected by chunk-block, min 3 common term pairs. Ranked by score$_0$						
POS filter	43.20±109.47	0.67±0.37	0.41±0.42	0.53	4.63±6.07	6.95±8.68
Standard	77.28±217.2	0.78±0.34	0.41±0.41	0.50	3.97±4.48	7.61±9.83
Cosine similarity among all the documents						
POS filter	All 2607	1.00	1.00	0.44	34.42±52.09	83.29±125.89
Standard	All 2607	1.00	1.00	0.44	27.58±47.45	74.41±107.60
Cosine similarity after the documents were selected by chunk-sentence, min 2 common term pairs						
POS filter	41.44±98.31	0.75±0.34	0.56±0.40	0.56	6.34±12.59	5.50±5.33
Standard	74.28±204.48	0.83±0.33	0.58±0.38	0.63	7.16±13.95	7.09±9.01

Table 1. Results of the initial tests.

Average rank. For term-pair matching, the average rank (i.e., average of the averages, see Section 2.3) of closely relevant documents lies between 2.16 and 4.66 for different document selection options. The average rank of related documents is lower than that of closely relevant documents, as expected.

Cosine similarity in the entire document collection demonstrates rather low average ranks – 27.58 and 34.42. If, however, we first filter the collection by term pairs and then apply cosine similarity to the documents that have survived the filtering, then the average ranks are comparable with those by score$_0$. For cosine similarity after document filtering, the average rank of related documents is higher than that of closely relevant documents, although it should be the other way around. One reason for that could be the low recall for related documents and fewer documents that drag the average rank down. Cosine similarity disregards the added value of term pairs that helps score$_0$ distinguish between closely relevant and related document.

P@1 lies around 0.5, which means that roughly each second top document is a correct answer. This is too little if we want to use our technique for fully automated email answering with a relevant webpage as the answer. Notably, cosine similarity yields the best P@1 value 0.63. Because the average ranks for cosine similarity have large standard deviation, cosine similarity offers higher gains (also better P@1) and bigger losses (also lower average rank) than score$_0$.

We selected the result typed boldfaced in Table 1 as the *best initial result and the baseline* for further experiments with term-pair matching: the best recall for closely relevant documents (0.83) and the second best recall for related documents (0.58), while having the third best and still good average ranks (3.47 and 6.51).

2.5 Amount of Matching Text

In order to better understand term-pair matching, we counted the number of term pairs and unique terms per query-document match for closely relevant documents for the best initial result in Table 1. Of the total 28 closely relevant documents, 24 were selected by the system during the test. Figure 1 shows the number of term pairs and unique terms from these term pairs for each of the 24 successful query-document matches. In most cases, a query-document match required no more than 6-7 term pairs; the largest number of term pairs was 26. The number of unique terms in these term pairs was also up to 6-7 per match, 13 at most. In eight matching cases, the number of terms in the matching term pairs was only 3.

On average, the terms in the term pairs accounted for about 40% of the unique query terms and about 3% of the unique document terms after compound splitting, lemmatization, and stop-word removal. That means that quite a large portion of the query but a small portion of the document participated in a successful query-document match.

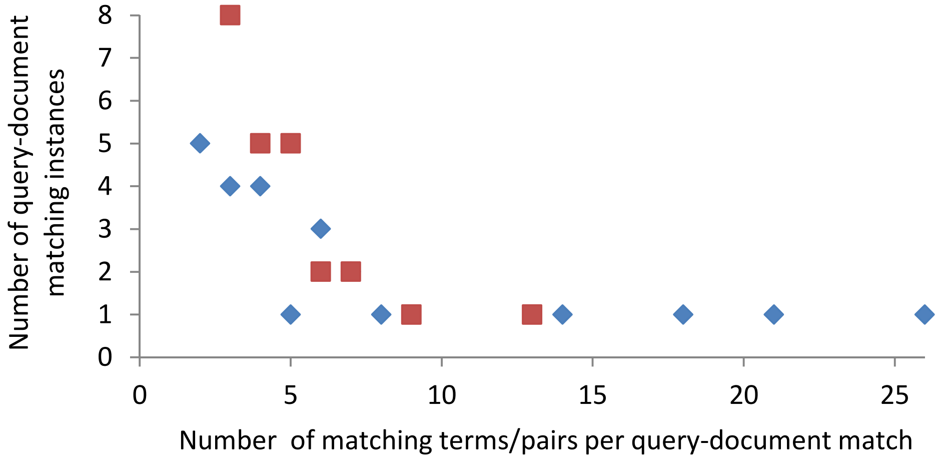
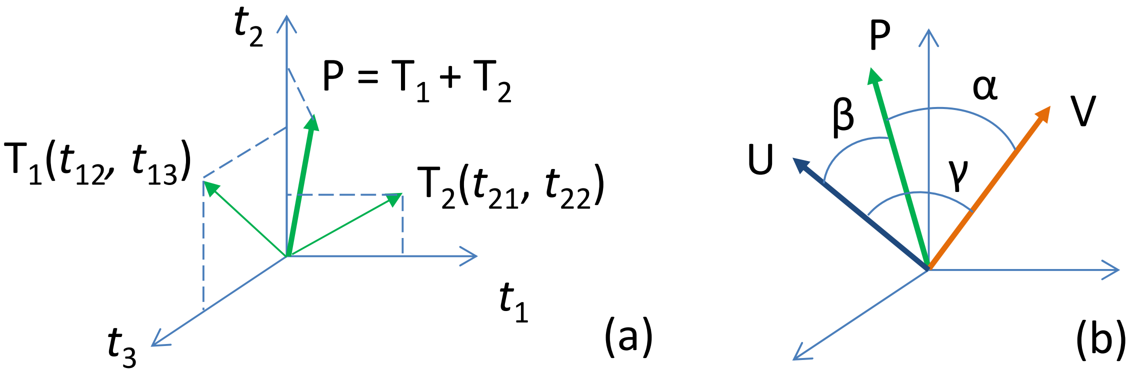


Figure 1. The number of term pairs (diamonds) and unique terms (squares) from these term pairs per query-document match, and the number of the corresponding matching instances. At (7; 2) a square hides a diamond, they overlap.

3 Text-Matching Model Based on Term Pairs in a Query-based Vector Space

While experimenting with the prototype, we designed formulas (1) to (6) according to our own perception of text relevance. Soon we realized that the formulas do have a representation in vector algebra. Let us imagine a term pair in the document *d* as a vector T_j; the term pair originates from one or several chunks of *d*. T_j has two coordinates $t_{ji} = w(t_i, \mathrm{T}_j, d)$ each calculated by formula (2). The document vector P is the sum of all the term-pair vectors T_j as shown in Figure 2 (a). The document vector P has coordinates $p_i = w(t_i, d)$ each calculated by formula (5).



t_2
P = T_1 + T_2
$\mathrm{T}_1(t_{12}, t_{13})$
$\mathrm{T}_2(t_{21}, t_{22})$
t_1
t_3
(a)
P
α
β
V
U
γ
(b)

Figure 2. (a) Document vector P as the sum of term-pair vectors. (b) Document vector P, vocabulary vector V, unigram vector U (used in Section 4), and the angles between them.

Traditionally in a vector space, the query and the document are two independent vectors. In Figure 2 we do not have any query vector; the query is not clearly visible. Still, the query is in there. Together with each individual document, the query establishes term pairs whose terms define the dimensions of the vector space. Furthermore, the query participates in the term weight formula (1) and, thus, contributes to the coordinates of P. The query provides the building blocks for the document vector P. *The query is embedded in the vector space and shapes the vector space*: limited dimensions of the vector space and query-term frequency in the coordinates of the term-pair vectors are the only presence of the query in the text-matching process.

How do we calculate similarity between the query, which is embedded in the vector space, and the document, which is a vector? One measure could be the length of P: longer P means more query participates in the document, which means more similarity. Another measure could be the volume of the polygon created by P: the more the document "fills" the query the more similar they are. We tested both measures; the average ranks were inferior to those of the best initial result in Table 1.

Let us examine why simple summation of vector coordinates works better than the length and volume scores. We can rewrite score_0 in formulas (5) and (6) as shown in formulas (7) and 8:

$$score_0(d) = \sum_{i=1}^{N} p_i \cdot 1 = P \bullet V = |P||V|\cos\alpha \qquad (7)$$

$$p_i = w(t_i, d) = tf(t_i, q)\cdot icf(t_i)\cdot \sum_{\{T\in d|t_i\in T\}} \left(icf(T)\cdot \sum_{\{ch\in d|T\in ch\}} tf(t_i, ch) \right) \qquad (8)$$

N is the number of dimensions of P, which is the number of unique terms in the term pairs. That number is not particularly large, as we see in Figure 1. V is the vocabulary vector, it has the same dimensions as P but the coordinates are 1. The vectors P and V and cos α in formula (7) are illustrated by Figure 2 (b). Score_0 favors the following query-document match conditions:

- | P | grows if p_i grows, i.e., (i) the query and the document have more term pairs in common and (ii) each term pair is more document-unique with higher inverted chunk frequency, (iii) each term in the term pairs has higher frequency in the document and (iv) in the query, and (v) the term is also more document-unique with higher inverted chunk frequency.
- $|V| = \sqrt{N}$, it grows along with the diversity of the vocabulary in the term pairs.

- cos α between vectors P and V is highest if all p_i values are the same, because vector V has each coordinate 1. Equal p_i values suggest that all terms in the term pairs should be equally important, where the importance is maintained by the balance between the frequency of the term in the query and the document, its uniqueness in the document collection, and uniqueness of its co-occurrences with other terms (i.e., term pairs). Less frequent terms require more unique co-occurrences.

In formula (7), the vocabulary vector V states that all terms in the term pairs are equal. Let us change it so that each coordinate of V is inverted chunk frequency of the term, as in formulas (9) and (10). Now the p_i values are not expected to be equal but instead be bigger for more unique terms in order to maintain good alignment between vectors P and V.

$$score_1(d) = P \bullet V = \sum_{i=1}^{N} p_i v_i \qquad (9)$$

$$v_i = icf(t_i) \qquad (10)$$

We tested formula (9) with the same options that our best initial result in Table 1 had: removal of standard stop-words, a chunk is a sentence, the query and the document must have at least 2 term pairs in common. Table 2 shows that score$_1$ yields better average ranks than the best initial result.

Formula	Avg P@1	Avg rank	
		Close	Related
score$_1$ (9)	0.50	3.38±3.93	6.24±6.81

Table 2. Query-document similarity score based on term pairs in a query-based vector space. (The corresponding number of selected documents and the recall values are boldfaced in Table 1.)

4 The Model Enhanced by Unigrams from the Document

Roughly 3% of the document terms participated in a successful query-document match, as stated in Section 2.5. Arguably, term co-occurrences in limited text chunks identify relevant parts of the document and use only these parts to score the document. Imagine an FAQ list. There are many FAQs and their answers in the document, but only one FAQ and its answer contains the right term co-occurrences. The system uses that FAQ to score the document and ignores the rest of the list.

What happens if we include all the vocabulary of the document into our query-document similarity score? If the document is concise, it may help. If we have a multi-topic document, the non-co-occurring and possibly not so relevant terms from the entire document are likely to damage the score. Let us try and see what happens.

Figure 2 (b) shows the new arrangement. We have the old vectors P and V, as well as a new unigram vector U, which is the traditional document term vector. For the highest score, P and V are aligned (α is small), as well as U and P are aligned (β is small), or U and V are aligned (γ is small). Good alignment between U and P or V requires that most document terms end up in the term pairs.

$$score_2(d) = (P \bullet V)\cos\beta = \sum_{i=1}^{N} p_i v_i \frac{\sum_{i=1}^{N} p_i u_i}{\sqrt{\sum_{i=1}^{N} p_i^2}\sqrt{\sum_{i=1}^{N} u_i^2}} \qquad (11)$$

$$score_3(d) = (P \bullet V)\cos\gamma = \sum_{i=1}^{N} p_i v_i \frac{\sum_{i=1}^{N} v_i u_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} u_i^2}} \qquad (12)$$

$$u_i = tf(t_i, d) \cdot tf(t_i, q) \cdot idf(t_i) \qquad (13)$$

Formulas (11), (12), (8), (10) and (13) implement this similarity score. Please observe that u_i – the coordinates of the unigram vector U – include the traditional term frequency in the entire document and inverted document frequency. No chunks. N here is the number of dimensions in the document, which means that many p_i and v_i values are 0; they are non-zero for the dimensions of the term pairs.

Formulas (8) and (13) include query-term frequency $tf(t_i, q)$ in the coordinates of vectors P and U. If we move $tf(t_i, q)$ from the coordinates of P and U to the coordinates of V, then U and V will start resembling the traditional document and query vectors. The dot product P • V will not change, the angles β and γ will. Let us redefine the coordinates p_i, v_i, and u_i as shown in formulas (14), (15), and (16). The new score$_4$ and score$_5$ are calculated as score$_2$ and score$_3$ but with p'_i, v'_i, and u'_i instead of p_i, v_i, and u_i.

$$p_i^{'} = icf(t_i) \cdot \sum_{\{T \in d | t_i \in T\}} \left(icf(T) \cdot \sum_{\{ch \in d | T \in ch\}} tf(t_i, ch) \right) \tag{14}$$

$$v_i^{'} = tf(t_i, q) \cdot icf(t_i) \tag{15}$$

$$u_i^{'} = tf(t_i, d) \cdot idf(t_i) \tag{16}$$

$$score_4(d) = score_2(d, p^{'}, v^{'}, u^{'}) \tag{17}$$

$$score_5(d) = score_3(d, p^{'}, v^{'}, u^{'}) \tag{18}$$

We tested the formulas with the same text processing and document selection options as in Section 3. Table 3 shows the results. In comparison with score$_1$, the average P@1 has improved in 3 of 4 cases. Notably, cosine similarity yielded the best P@1 in Table 1; now cosine similarity is somewhat smuggled in by the use of cos β and cos γ. As to the average ranks of closely relevant documents, score$_1$ with only term pairs and without the unigram vector U had a marginally better value. For related documents, vector U was beneficial for score$_4$ and score$_5$ where query-term frequency was in the coordinates of vector V.

Formula	Avg P@1	Avg rank	
		Close	Related
score$_2$ (11)	0.56	3.44±4.17	6.18±6.84
score$_3$ (12)	0.44	3.66±4.04	6.06±7.51
score$_4$ (17)	0.56	4.16±4.40	4.61±4.30
score$_5$ (18)	0.56	3.56±3.26	4.60±3.98

Table 3. Query-document similarity based on term pairs and unigrams from the document. (The corresponding number of selected documents and the recall values are boldfaced in Table 1.)

5 Advantages and Limitations

Our term-pair-based text-matching technique has three *advantages*. First of all, it has a good average rank for closely relevant messages; we measured 3.38±3.93 at average recall 0.83±0.33. The second advantage is low maintenance costs, because the technique does not have any knowledge base to develop and maintain. Finally, we may argue that term co-occurrence identifies relevant parts of the document and ignores the rest (only about 3% of the document terms are used), which means the technique works pretty well with multi-topic documents.

The first *limitation* is email-size queries. The query is big enough to shape a vector space and small enough to be one text chunk. We have not yet considered this technique for other query sizes. Furthermore, we know that a message sent to a contact center is keyword-rich text because it describes the context and requires a response (Sneiders, Sjöbergh and Alfalahi, 2016). We do not know how the technique would work with queries where keyword density is lower than that of email-style messages.

The design of the technique had a question-answering task in mind. In our experiments, term-pair matching worked best for closely relevant documents. Topic-related text retrieval and categorization, which most of text retrieval is, does not deal with the concept of answer.

Because a successful query-document match relies on rather few term pairs (see Section 2.5), the technique relies on good text pre-processing tools – spelling correction, compound splitting, lemmatization – which lay the groundwork for a larger number of term pairs.

The average P@1 value around 0.5 is nice but not sufficient for fully automated email answering.

6 Related and Future Research

Utilizing term co-occurrence in a text chunk is not a new phenomenon in email answering at contact centers. Malik at al. (2007) match sentences in a query message to tag-questions (like FAQs) attached to standard answers. Marom and Zukerman (2005) cluster answers given to similar inquiries, and then pick out most representative sentences from the answers in one cluster in order to build a model answer. Sneiders, Sjöbergh and Alfalahi (2016) match manual text patterns to email messages for assigning standard answers. A text pattern looks for co-occurring terms in a paragraph of the query; a phrase in the text pattern considers the sequence and distance between the terms in a sentence.

Corpus-based question answering (QA) relies on term co-occurrence in a limited piece of text. Let us consider one of the earlier works (Kwok et al., 2001) which pinpoints the main steps of QA. Retrieval of source documents for candidate answers requires good search queries, which means a good set of co-occurring terms. Answer extraction involves locating regions in the documents with high density of the search keywords. Finally, answer selection calculates the weight of each candidate answer, which is a text snippet, based on co-occurrence of many and relevant terms in the snippet.

Term co-occurrence in a document (not a chunk) has been used to build new text categorization features, along with the unigrams (Figueiredo et al, 2011). The logic there is the same as ours – term co-occurrence creates new meaning that is not visible if we consider each term separately.

Proximity of the query terms in a document has been used to enhance the probabilistic retrieval model in BM25 for web search (e.g., He et al., 2011; Svore et al., 2010). Song at al. (2008) identify text spans that contain query terms in the document, and calculate relevance contribution of a span with respect to each query term in there. The relevance contribution is used instead of term frequency in the BM25 document score.

As for our own text-matching model, it needs tests with more data and more baseline systems in order to see whether the perceived value of the model is real.

Synonyms and semantic distance between terms is something to test. They would increase the number of both relevant and non-relevant term pairs. Domain-specific synonyms are likely to help, but that would mean developing a “knowledge base”, something we wanted to avoid in the beginning.

Sneiders, Sjöbergh and Alfalahi (2016) have observed that the essence of the message comes in one sentence in half of the email messages sent to a contact center. If we learn to identify that sentence, then we can greatly reduce the number of non-relevant term pairs coming from the rest of the message.

So far we did not consider co-occurrence of more than 2 terms in a combination – a triple is two pairs. Whether the added value of a new term in the combination of terms is linear (the sum of pairs) or not, we do not know it yet.

Previously we argued that term-pair matching identifies the most relevant parts of a multi-topic document and calculates query-document similarity with respect to those parts. Currently the entire document is retrieved to the user the way search engines do, whereas a better option might be extracted relevant parts of the multi-topic document.

7 Conclusion

Traditionally in text retrieval, presence of terms in a document is the main information carrier. We have shown that term co-occurrence in a limited text chunk (a sentence or paragraph) is an information carrier in its own right. In our experiments, only about 3% of unique document terms were used to successfully match the document to the query (Section 2.5). These 3% came from the term pairs that text chunks in the document had in common with the query.

Table 1 shows the success of our term pair matching for document *filtering*. For example, the selected 74 (on average) out of 2607 documents yielded 83% (on average) recall for the closely relevant documents. Ranking among a few selected documents with high density of relevant documents is more effective than ranking in the entire document collection.

Furthermore, we used the term pairs to score the documents for *relevance ranking*. At the 83% recall, the system could reach the average rank of closely relevant documents 3.38 (Table 2) while cosine similarity in the same settings yielded 7.16 (Table 1).

Our main academic contribution is a simple but effective *text-matching model* where the query shapes the vector space and a document is modelled by two or three vectors. (i) The document vector is the sum of individual term-pair vectors; it favors many and subject-specific term pairs with subject-relevant terms. (ii) The vocabulary vector favors a large diversity of vocabulary in the term pairs and equally important terms in the term pairs. (iii) The optional unigram vector favors matches where most of the document terms end up in the term pairs. The model is easy to visualize.

Reference

Cory Barr, Rosie Jones and Moira Regelson. 2008. The Linguistic Structure of English Web-Search Queries. Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, 1021-1030.

Johan Carlberger and Viggo Kann. 1999. Implementing an efficient part-of-speech tagger. Software-Practice and Experience, 29(9):815-32.

Fábio Figueiredo, Leonardo Rocha, Thierson Couto, Thiago Salles, Marcos André Gonçalves and Wagner Meira Jr. 2011. Word co-occurrence features for text classification. Information Systems, 36(5):843-858.

Ben He, Jimmy Xiangji Huang and Xiaofeng Zhou. 2011. Modeling term proximity for probabilistic information retrieval models. Information Sciences, 181(14), pp.3017-3031.

Cody Kwok, Oren Etzioni and Daniel S. Weld. 2001. Scaling question answering to the web. ACM Transactions on Information Systems (TOIS), 19(3):242-262.

Rahul Malik, L. Venkata Subramaniam and Saroj Kaushik. 2007. Automatically Selecting Answer Templates to Respond to Customer Emails. IJCAI, Vol. 7, 1659-1664.

Yuval Marom and Ingrid Zukerman. 2005. Towards a framework for collating help-desk responses from multiple documents. Proceedings of the IJCAI05 Workshop on Knowledge and Reasoning for Answering Questions, 32-39.

Mozilla. 2016. Block-level elements. https://developer.mozilla.org/en/docs/Web/HTML/Block-level_elements Accessed in July 2016.

Stephen E. Robertson and K. Sparck Jones. 1976. Relevance weighting of search terms. Journal of the American Society for Information Science, 27(3):129-146.

Gerard Salton and Michael J. McGill. 1986. Introduction to Modern Information Retrieval. McGraw-Hill, Inc. New York, NY, USA.

Jonas Sjöbergh and Viggo Kann. 2004. Finding the Correct Interpretation of Swedish Compounds, a Statistical Approach. Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC), 899–902.

Eriks Sneiders, Gunnar Eriksson and Alyaa Alfalahi. 2014. Exploring the Traits of Manual E-Mail Categorization Text Patterns. Advances in Natural Language Processing. Springer International Publishing, 337-344.

Eriks Sneiders. 2016. Review of the Main Approaches to Automated Email Answering. Advances in Intelligent Systems and Computing. Springer International Publishing, 135-144.

Eriks Sneiders, Jonas Sjöbergh and Alyaa Alfalahi. 2016. Email Answering by Matching Question and Context-Specific Text Patterns: Performance and Error Analysis. Advances in Natural Language Pro-cessing. Springer International Publishing, 123-133.

Ruihua Song, Michael J. Taylor, Ji-Rong Wen, Hsiao-Wuen Hon and Yong Yu. 2008. Viewing term proximity from a different perspective. Advances in information retrieval (proceedings of ECIR 2008). Springer Berlin Heidelberg, 346-357.

Krysta M. Svore, Pallika H. Kanani and Nazan Khan. 2010. How Good is a Span of Terms? Exploiting Proximity to Improve Web Retrieval. Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. ACM, 154-161.

Text Tools. 2016. Swedish stop words. http://countwordsfree.com/stopwords/swedish . Accessed in July 2016.