Assignment 6

Mousumi Akter

April 2020

1 Information Retrieval Evaluation

a. Simply using raw term frequency counts with a dot product similarity might not produce the best ranking results. Because "stopwords" always have a higher frequency in the documents. Only using term frequency doesn't help us to distinguish between a content word like "food" from a stopword like "about".

b. Before adding d the IDF is:
$IDF(w) = log(\frac{M+1}{df(w)})$
where M is the total number of documents in the collection and df(.) counts the document frequency (the total number of documents containing w).
After adding d the IDF is (Considering word w also appears in doc d):
$IDF(w) = log(\frac{M+2}{df(w)+1})$
So, the change of IDF is:
$\Delta IDF(w) = log(\frac{M+1}{df(w)}) - log(\frac{M+2}{df(w)+1})$

d.

Rank	Relevance	Precision	Recall	F1-Score
1	+	1/1	1/16	0.1176
2	+	2/2	2/16	0.2222
3	-	2/3	2/16	0.2105
4	+	3/4	3/16	0.3
5	+	4/5	4/16	0.381
6	-	4/6	4/16	0.3636
7	-	4/7	4/16	0.3478
8	+	5/8	5/16	0.4167
9	-	5/9	5/16	0.4
10	-	5/10	5/16	0.3846

Now, calculating average precision:

Mathematically, we can define average precision on a ranked list L where $|L| = n$ as

$$avp(L) = \frac{1}{|Rel|} \sum_{i=1}^{n} p(i)$$

where p(i) denotes the precision at rank i of the documents in L, and Rel is the set of all relevant documents in the collection. If D_i is not relevant, we would ignore the contribution from this rank by setting $p(i) = 0$. If D_i is relevant, to obtain p(i) we divide the number of relevant documents we've seen so far by the current position in the list (which is i).

i	Relevance	p(i)
1	+	1/1
2	+	2/2
3	-	0
4	+	3/4
5	+	4/5
6	-	0
7	-	0
8	+	5/8
9	-	0
10	-	0

$sum = 4.175$
$avp = \frac{4.175}{16} = 0.2609$

e. i. Cumulative Gain at 7 documents

Rank	Relevance	Gain	Cumulative Gain
1	+	1	1
2	+	1	1+1=2
3	-	0	1+1+0=2
4	+	1	1+1+0+1=3
5	+	1	1+1+0+1+1=4
6	-	0	1+1+0+1+1+0=4
7	-	0	1+1+0+1+1+0+0=4

So, the CumulativeGain@7 = 4

ii. Normalized Discounted Cumulative Gain at 7 documents

Rank	Relevance	Gain	Cumulative Gain	Discounted Cumulative Gain
1	+	1	1	1
2	+	1	2	$1+1/\log_2 2=2$
3	-	0	2	$1+1/\log_2 2+0=2$
4	+	1	3	$1+1/\log_2 2+0+1/log_2 4=2.5$
5	+	1	4	$1+1/\log_2 2+0+1/log_2 4+1/log_2 5=2.93$
6	-	0	4	$1+1/\log_2 2+0+1/log_2 4+1/log_2 5+0=2.93$
7	-	0	4	$1+1/\log_2 2+0+1/log_2 4+1/log_2 5+0+0=2.93$

DCG@7 = 2.93

IdealDCG@7 = $1+1/\log_2 2+1/\log_2 3+1/\log_2 4+0/\log_2 5+0/\log_2 6+0/\log_2 7$

$= 1 + 1/\log_2 2 + 1/\log_2 3 + 1/\log_2 4$

$= 3.1309$

Now, NormalizedDCG = $\frac{DCG@7}{IdealDCG@7} = \frac{2.93}{3.1309} = 0.935$

2 Word Embeddings (Theory)

a. What is the distributional hypothesis and how is it incorporated into Skip-Gram and CBOW?
Ans: The distribution hypothesis is:
Words that are high similar occur in the same contexts as one another.
The distribution hypothesis incorporated into Skip-Gram because a word ought to be able to predict its context. For an example, if word "University" is used it should predict word like "Professor, Lecture, Class, Exam, Assignments, Research etc."
The distribution hypothesis incorporated into CBOW because a context ought to be able to predict its missing word. For an example, if a sentence is said "Professor gives — in the University", the missing word "Lecture" should be predicted.

b. The actual data is the positive examples of the neighboring word.
The noise distribution is the negative examples of the neighboring words.
Now, how does that negative examples are selected. It's a free parameter in word2vec and in the paper it is mentioned that the unigram distribution U(w) raised to the 3/4rd power (i.e., $\frac{U(w)^{3/4}}{Z}$) outperforms significantly. Now my understanding is that it omits super frequent word like "the" but takes word in the more or less middle frequency range as a negative example.

c. i. How does the subsampling affect very frequent and infrequent words? How does the subsampling affect the ranking of words when ordered by frequency?
Ans: Based on my understanding, the goal of subsampling is to reduce the number of times we sample frequent words. Because as mentioned in the paper, the vector representations of frequent words do not change significantly after training on several million examples. In the formulae, $f(w_i)$ as a number between 0 and 1, i.e., the number of times word w_i appears divided by the total number of words. For a very frequent word (appears as a 10^{-3} fraction of the words), we have $p(w_i) = 1 - \sqrt{10^{-2}} = 0.9$, so it's very likely to be discarded. For a moderately infrequent word (probability of occurrence 10^{-5}), then we have $p(w_i) = 1 - \sqrt{1} = 0$, so it won't be discarded. For a word that is even rare than that, it won't be discarded either.
So this discards very frequent words with high probability and keeps almost all words that are rare or moderately rare.

ii. How does the subsampling affect the size of the context window? Ans: One important implementation detail of subsampling in word2vec is that the random removal of tokens is done before the corpus is processed into word-context pairs (i.e. before word2vec is actually run). This essentially enlarges the context window's size for many tokens, because they can now reach words that were not in their original L-sized windows. For example, "the quick brown fox jumped over the lazy dog" might be reduced to "quick brown fox jumped lazy dog". In this case, "dog" is now in the 3-word context window of "jumped", even though it was originally 4 tokens away from "jumped".

3 Word Embeddings (Practice)

a. dog, mouse, data.

b. data, mouse, also.

c. also, it, for.

d. cluster1=[the, it, for, also] as they are stopwords
cluster2=[cat, dog, mouse] as they are word related to animal
cluster3=[computer, data] as they are word related to computer

e. Yes, I see a problem on hard clustering. Suppose, if we take 3-nearest neighbor then word "data" goes to cluster of word related to animal which doesn't make sense.

4 KL-divergence Retrieval Function

a. Show that the KL-divergence retrieval function covers the query likelihood retrieval function as a special case if we set the query language model to the empirical word distribution in the query.
Ans:

Query likelihood
$$f(q,d) = \sum_{\substack{w \in d \\ w \in q}} \boxed{c(w,q)} \left[\log \frac{p_{seen}(w|d)}{\alpha_d p(w|C)}\right] + n \log \alpha_d$$

KL-divergence (cross entropy)
$$f(q,d) = \sum_{w \in d, p(w|\theta_Q) > 0} \boxed{[p(w|\hat{\theta}_Q)} \log \frac{p_{seen}(w|d)}{\alpha_d p(w|C)}] + \log \alpha_d$$

Query LM
$$p(w|\hat{\theta}_Q) = \frac{c(w,Q)}{|Q|}$$

The difference between original query likelihood formula and the generalized KL-divergence model is only the empirical word distribution in the query. On top, we have the query likelihood retrieval function. The KL-divergence retrieval model generalizes the query term frequency into a probabilistic distribution. This distribution is the only difference, which is able to characterize the user's query in a more general way. This query language model can be estimated in many different ways—including using feedback information. This method is called KL-divergence because this can be interpreted as measuring the divergence (i.e., difference) between two distributions; one is the query model $p(w|\theta_Q)$ and the other is the document language model from before.