Off the Beaten Path: Let’s Replace Term-Based Retrieval with $k$-NN Search

Leonid Boytsov, David Novak, Yury Malkov, Eric Nyberg

CIKM 2016
Where we stand

NLP

Our work

IR

k-NN
Motivation

• **Our belief:** Adhoc text retrieval could and should benefit from using **generic** $k$-NN search algorithms

• **Common belief:** $k$-NN search is **horribly** slow
Classic Text Retrieval: Overview

- Classic text retrieval employs a filter-and-refine pipeline
- The list of candidates is generated using a simple TF×IDF ranking function
- Sophisticated similarity is used only for re-ranking
- Filtering errors cannot be fixed downstream
- Many filtering errors stem from a vocabulary mismatch

Zhao and Callan (2010); Furnas et al. (1987)
Our Conjectures

• Incorporating sophisticated similarity features into early retrieval stages may facilitate:
  • finding of relevant documents that term-based methods cannot retrieve;
  • a separation of labor between data scientists (focusing on modelling) and software engineers (focusing on efficiency).

• Generic $k$-NN search could be fast and practical.
CURSE OF DIMENSIONALITY

Source: http://liledekahan.eklablog.com/tuhes-paysages-mystiques-c176638
CURSE OF DIMENSIONALITY

Source: http://liledekahlan.eklablog.com/tuiles-paysages-mystiques-c176638
Challenges of Applying $k$-NN Search
Challenges of Applying $k$-NN Search
Which model to use?

- BM25 + IBM Model 1
  20% more effective than BM25 (Surdeanu et al 2011)

- Cosine similarity over averaged word embeddings
  State-of-the-art in NLP + efficient search algorithms
IBM Model 1

- IBM Model 1 is a lexical translation model
- Possible training data: question answer pairs
- Learns associations between query and documents terms
- First application in IR by Berger and Lafferty (1999)
### IBM Model 1: Sample Translation Probabilities

The table below shows the probability $T(\text{question term} | \text{answer term})$ for Yahoo Answers:

<table>
<thead>
<tr>
<th>Question term</th>
<th>Answer term</th>
<th>$T$</th>
<th>Answer term</th>
<th>$T$</th>
<th>Answer term</th>
<th>$T$</th>
<th>Answer term</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>caffeine</td>
<td>coffee</td>
<td>0.074</td>
<td>drink</td>
<td>0.051</td>
<td>tea</td>
<td>0.021</td>
<td>caffeine</td>
<td>0.018</td>
</tr>
<tr>
<td>latte</td>
<td>milk</td>
<td>0.052</td>
<td>coffee</td>
<td>0.049</td>
<td>starbucks</td>
<td>0.043</td>
<td>espresso</td>
<td>0.021</td>
</tr>
<tr>
<td>oxygen</td>
<td>o2</td>
<td>0.043</td>
<td>air</td>
<td>0.024</td>
<td>gas</td>
<td>0.018</td>
<td>carbon</td>
<td>0.018</td>
</tr>
<tr>
<td>contagious</td>
<td>spread</td>
<td>0.024</td>
<td>person</td>
<td>0.022</td>
<td>infection</td>
<td>0.013</td>
<td>virus</td>
<td>0.013</td>
</tr>
<tr>
<td>likelihood</td>
<td>likely</td>
<td>0.015</td>
<td>chance</td>
<td>0.015</td>
<td>chances</td>
<td>0.013</td>
<td>odds</td>
<td>0.013</td>
</tr>
<tr>
<td>animal</td>
<td>animals</td>
<td>0.110</td>
<td>dog</td>
<td>0.037</td>
<td>dogs</td>
<td>0.019</td>
<td>cat</td>
<td>0.017</td>
</tr>
<tr>
<td>study</td>
<td>studying</td>
<td>0.037</td>
<td>school</td>
<td>0.021</td>
<td>studies</td>
<td>0.018</td>
<td>exam</td>
<td>0.013</td>
</tr>
<tr>
<td>challenge</td>
<td>like</td>
<td>0.016</td>
<td>challenges</td>
<td>0.015</td>
<td>think</td>
<td>0.012</td>
<td>all</td>
<td>0.010</td>
</tr>
<tr>
<td>fat</td>
<td>weight</td>
<td>0.061</td>
<td>eat</td>
<td>0.027</td>
<td>lose</td>
<td>0.024</td>
<td>body</td>
<td>0.023</td>
</tr>
</tbody>
</table>
**IBM Model 1 : Sample Translation Probabilities**

Probability $T(\text{question term}|\text{answer term})$ for Yahoo Answers:

<table>
<thead>
<tr>
<th>Question term</th>
<th>Answer term</th>
<th>Answer term</th>
<th>Answer term</th>
<th>Answer term</th>
</tr>
</thead>
<tbody>
<tr>
<td>caffeine</td>
<td>coffee</td>
<td>drink</td>
<td>tea</td>
<td>caffiene</td>
</tr>
<tr>
<td>latte</td>
<td>milk</td>
<td>coffee</td>
<td>starbucks</td>
<td>espresso</td>
</tr>
<tr>
<td>oxygen</td>
<td>o2</td>
<td>air</td>
<td>gas</td>
<td>carbon</td>
</tr>
<tr>
<td>contagious</td>
<td>spread</td>
<td>person</td>
<td>infection</td>
<td>virus</td>
</tr>
<tr>
<td>likelihood</td>
<td>likely</td>
<td>chance</td>
<td>chances</td>
<td>odds</td>
</tr>
<tr>
<td>animal</td>
<td>animals</td>
<td>dog</td>
<td>dogs</td>
<td>cat</td>
</tr>
<tr>
<td>study</td>
<td>studying</td>
<td>school</td>
<td>studies</td>
<td>exam</td>
</tr>
<tr>
<td>challenge</td>
<td>like</td>
<td>challenges</td>
<td>think</td>
<td>all</td>
</tr>
<tr>
<td>fat</td>
<td>weight</td>
<td>eat</td>
<td>lose</td>
<td>body</td>
</tr>
</tbody>
</table>

12/26 C +-2016
Probability that query $Q$ is a translation of document $D$:

$$P(Q|D) = \prod_{q \in Q} P(q|D)$$

$$P(q|D) = (1 - \lambda) \left[ \sum_{d \in D} T(q|d)P(d|D) \right] + \lambda P(q|C)$$

$$T(q|d) = T(\text{question term}|\text{document term})$$
IBM Model 1

Expensive to compute: \(|Q| \times |D|\) hash table lookups

Probability of a translation of document \(D\):

\[
P(Q|D) = \prod_{q \in Q} P(q|D)
\]

\[
P(q|D) = (1 - \lambda) \left[ \sum_{d \in D} T(q|d)P(d|D) \right] + \lambda P(q|C)
\]

\[
T(q|d) = T(\text{question term}|\text{document term})
\]
Which $k$-NN Algorithms to Use?

- A proximity-graph algorithm Small-World graph (SW-graph)
  Malkov et al. 2014+ later improvements by Malkov

- A pivoting algorithm Neighborhood APProximation index (NAPP)
  Tellez et al. (2013)
What Worked for Which Data

- SW-graph works well for averaged word embeddings
- NAPP works well only for similarities on sparse representations ...
What Worked for Which Data

- SW-graph works well for averaged word embeddings
- NAPP works well only for similarities on sparse representations . . .
- ...but it requires a special set of generated pivots
  Pivots are “built” from randomly selected terms (David Novak)
Test Collections for Adhoc Retrieval

- Yahoo Answers and Stack Overflow
- Use questions and respective best answers
- Find answers using only the question text
- Train IBM Model 1 on a subset that does not overlap with train/dev/test data
Overview of Results

Efficiency-effectiveness Trade-Offs of $k$-NN Search (lower and to the right is better)
Main Conclusions

- Generic $k$-NN search can be efficient and accurate in text retrieval
- It can be beneficial to employ sophisticated similarity at an early retrieval stage
Three Futures of $k$-NN in Adhoc Retrieval

- Purely sparse representations (bag of words and/or entities)
  Work better for adhoc search

- Purely dense representations
  Work better for collaborative filtering

- Hybrid approaches
  May work well if we could improve dense similarity models
Thank you for attention! Questions?

Our code is on GitHub:


It relies on the efficient library for $k$-NN search:

https://github.com/searchivarius/nmslib


Pipeline Architecture

Retrieval modules

- **$k$-NN (NMSLIB)**
- SW-graph
- Brute force

Term-based (Lucene)
- Inverted index

Term Lists
- In-memory forward index

Optional re-ranker
- $k=500$ candidates
- $k=100$ candidates

question

Cosine Embed

original words

Cosine TF×IDF lemmas

BM25 lemmas

BM25+Model 1 lemmas

Brute force

NAPP

answer recall

$P@1, MRR$

$R@k$

highest-score answer

$k=N$ answers

$N$ answers

$k=N$ candidates
Replacing The Original Function to Compute Distance to Pivots

**Comprehensive**

**BM25+Model 1**

- BM25
- BM25+Model 1

**BM25**

- BM25
- TF×IDF cosine

---

23/26 CIKM 2016
Replacing The Original Function to Compute Distance to Pivots

Stack Overflow

BM25+Model 1

BM25

改善在距离计算的数量

BM25

BM25+Model 1

TF×IDF cosine

0.86 0.88 0.9 0.92 0.94 0.96 0.98

0 100 200 300 400

R@1

改进在距离计算的数量

BM25

TF×IDF cosine

0.86 0.88 0.9 0.92 0.94 0.96 0.98

0 100 200 300 400

R@1

24/26
The answer recall at different $N$

**Comprehensive**

![Graph showing answer recall vs. result set size $N$ (log scale)]

- BF $bm25+model1$
- NAPP $bm25+model1$
- BF $bm25$
- BF cosine embed
- BF cosine tf-idf +rerank
- Lucene
- NAPP $bm25$
- SW-graph cosine embed
- BF cosine tf-idf -rerank
The answer recall at different $N$

**Stack Overflow**

The graph shows the answer recall at different result set sizes $N$ (log scale) for various methods:

- BF bm25+model1
- NAPP bm25+model1
- BF bm25
- Lucene
- NAPP bm25
- BF cosine tf-idf +rerank
- BF cosine embed
- SW-graph cosine embed
- BF cosine tf-idf -rerank

The graph illustrates how the recall changes as the result set size increases for each method.