Flexible Similarity Search of Semantic Vectors
Using Fulltext Search Engines

Michal Růžička, Vít Novotný, Petr Sojka; Jan Pomikálek, Radim Řehůřek

Masaryk University, Faculty of Informatics, Brno, Czech Republic
mruzicka@mail.muni.cz, witiko@mail.muni.cz, sojka@fi.muni.cz;
RaRe Technologies
honza@rare-technologies.com, radim@rare-technologies.com


Illustrations by Jiří Franek.
Outline

1. Semantic Indexing and Searching
2. String Encoding of Semantic Vectors
3. Results
Outline

1. Semantic Indexing and Searching

2. String Encoding of Semantic Vectors

3. Results
Semantic Indexing

**Input Document**

*Document as a file (e-mail, PDF, ...), URL, ...*

**DataReader**

(e.g. pdf2text)

*Document as plain text*

**Tokenizer**

(e.g. NLTK tokenizer)

**Segment2Vec**

(e.g. TFIDF, LSI, deep learning, doc2vec)

*Document as a segment list*

**SemanticModeler**

*Segments in all documents*

**Segmenter**

(e.g. paragraph / logical part [table, formula] segmenter)

*Document as a token list*

**Index of Vectors**

*Document as a list of points representing segments*
Semantic Searching with Nuggets

- **Query Document**
- **Indexing Pipeline**
- **Document Nuggets**
- **Query Nuggets**
- **Similarity Search**
  - $q \cdot K$ semantic vectors $q = 3, K = 2$
- **Candidate Nuggets**
- **Ranker**
  - $k = 3$
- **Results as Sorted Nuggets**
- **Results as Sorted Documents**
Re-Ranking Techniques

1. **Fast**: find candidate nuggets via Elasticsearch.

2. **Slow but precise**: re-rank candidate nuggets with exact similarity metric.
   - Cosine similarity.
   - Euclidean similarity.
Re-Ranking Techniques

1. **Fast**: find candidate nuggets via Elasticsearch.

2. **Slow but precise**: re-rank candidate nuggets with exact similarity metric.
   - Cosine similarity.
   - Euclidean similarity.
   - ...

Flexible Similarity Search of Semantic Vectors Using Fulltext Search Engines

Outline

1. Semantic Indexing and Searching

2. String Encoding of Semantic Vectors

3. Results
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \[ \vec{w} = [0.12, -0.13, 0.065] \]
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \[ \vec{w} = [0.12, -0.13, 0.065] \]
  - Rounding to two decimal places, string encoded:
    \[ \vec{\hat{w}} = [0.12, -0.13, 0.065] \]
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    $$\mathbf{w} = [0.12, -0.13, 0.065]$$
  - Rounding to two decimal places, string encoded:
    $$\mathbf{\hat{w}} = ['0' 0.12, -0.13, 0.065]$$
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \[ \mathbf{\vec{w}} = [0.12, -0.13, 0.065] \]
  - Rounding to two decimal places, string encoded:
    \[ \mathbf{\vec{\hat{w}}} = ['0' 0.12, '1' -0.13, 0.065] \]
String Encoding of Semantic Vectors

• Encoding of semantic vectors to strings (feature tokens):
  • Semantic vector of three dimensions:
    \[ \mathbf{\hat{w}} = [0.12, -0.13, 0.065] \]
  • Rounding to two decimal places, string encoded:
    \[ \mathbf{\hat{w}} = ['0' 0.12, '1' -0.13, '2' 0.065] \]
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \[ \mathbf{w} = [0.12, -0.13, 0.065] \]
  - Rounding to two decimal places, string encoded:
    \[ \hat{\mathbf{w}} = ['0P2', '1', '2'] \]

Feature tokens:
- 0P2i0d12
- 1P2ineg0d13
- 2P2i0d07
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \[ \vec{w} = [0.12, -0.13, 0.065] \]
  - Rounding to two decimal places, string encoded:
    \[ \vec{\hat{w}} = ['0P2', 0.12, '1P2', -0.13, '2', 0.065] \]
String Encoding of Semantic Vectors

• Encoding of semantic vectors to strings (feature tokens):
  • Semantic vector of three dimensions:
    \[ \mathbf{\hat{w}} = [0.12, -0.13, 0.065] \]
  • Rounding to two decimal places, string encoded:
    \[ \mathbf{\hat{w}} = ['0P2' 0.12, '1P2' -0.13, '2P2' 0.07] \]
String Encoding of Semantic Vectors

• Encoding of semantic vectors to strings (feature tokens):
  • Semantic vector of three dimensions:
    \[ \mathbf{w} = [0.12, -0.13, 0.065] \]
  • Rounding to two decimal places, string encoded:
    \[ \hat{\mathbf{w}} = ['0P2i0d12', '1P2' -0.13, '2P2' 0.07] \]
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \[ \mathbf{\hat{w}} = [0.12, -0.13, 0.065] \]
  - Rounding to two decimal places, string encoded:
    \[ \mathbf{\hat{w}} = ['0P2i0d12', '1P2ineg0d13', '2P2' 0.07] \]
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \[ \mathbf{\hat{w}} = [0.12, -0.13, 0.065] \]
  - Rounding to two decimal places, string encoded:
    \[ \mathbf{\hat{w}} = ['0P2i0d12', '1P2ineg0d13', '2P2i0d07'] \]
String Encoding of Semantic Vectors

- Encoding of semantic vectors to strings (feature tokens):
  - Semantic vector of three dimensions:
    \( \mathbf{\hat{w}} = [0.12, -0.13, 0.065] \)
  - Rounding to two decimal places, string encoded:
    \( \mathbf{\hat{w}} = ['0P2i0d12', '1P2ineg0d13', '2P2i0d07'] \)
  - Feature tokens:
    - 0P2i0d12
    - 1P2ineg0d13
    - 2P2i0d07
High-Pass Filtering – Speed Optimization

- High-pass filtering: \( \vec{w} = [0.12, -0.13, 0.065] \)

  **trim**  Fixed threshold, for example 0.1:
  Keep only 0.12, \(-0.13\) from \(\vec{w}\), as \(|0.065| < 0.1\).

  **best**  Fixed number of the best values is used, for example only the best one:
  Keep only \(-0.13\) from \(\vec{w}\), as \(|-0.13|\) is the highest absolute value in \(\vec{w}\).
High-Pass Filtering – Speed Optimization

- High-pass filtering: \( \vec{w} = [0.12, -0.13, 0.065] \)

  **trim** Fixed threshold, for example 0.1:
  Keep only 0.12, \(-0.13\) from \(\vec{w}\), as \(|0.065| < 0.1\).

  **best** Fixed number of the best values is used, for example only the best one:
  Keep only \(-0.13\) from \(\vec{w}\), as \(|-0.13|\) is the highest absolute value in \(\vec{w}\).

Speed optimization of the search for candidate nuggets

*without significant impact on the quality.*
Outline

1. Semantic Indexing and Searching
2. String Encoding of Semantic Vectors
3. Results
Datasets

**en-wiki**  The English Wikipedia dataset.
- **LSA** with 400 dimensions
- **doc2vec** with 400 dimensions.

**wiki-2014+gigaword-5**  Pre-trained word vectors from Wikipedia and English Gigaword Fifth Edition.
- **GloVe** with 50, 100, 200, and 300 dimensions.

**common-crawl**  Pre-trained word vectors from the Common Crawl project.
- **GloVe** with 300 dimensions.

**twitter**  Pre-trained word vectors from the Twitter social network.
- **GloVe** with 25, 50, 100, and 200 dimensions.

**texmex**  Image descriptors provided by the TEXMEX project.
- **SIFT** descriptors of images with 128 dimensions.
Comparison of Results

- **English Wikipedia Cosine Similarity**
- **TEXMEX SIFT Descriptors Cosine Similarity**
- **TEXMEX SIFT Descriptors Euclidean Similarity**
Comparison of Results

![Graph showing comparison of results for English Wikipedia Cosine Similarity and Cosine Similarity with different page sizes and number of best features used. The graph illustrates how precision increases with page size and decreases as fewer features are used.](image-url)
Comparison of Results

![Graph showing the comparison of results for different page sizes and feature usages. The graph plots Precision@10 on the y-axis against Page size on the x-axis. The lines represent different numbers of best features used: all (purple), 90 (red), 40 (green), 17 (orange), and 6 (blue). The graph compares TEXMEX SIFT Descriptors Cosine Similarity and Cosine Similarity.]
Comparison of Results

![Graph showing comparison of results between TEXMEX SIFT Descriptors and Euclidean Similarity. The graph plots Precision@10 against page size. Different line colors represent the number of best features used: all, 90, 40, 17, and 6.](image-url)
Comparison of Results

English Wikipedia
Cosine Similarity

 TEXMEX SIFT Descriptors
Cosine Similarity

 TEXMEX SIFT Descriptors
Euclidean Similarity
Summary

Flexible: Different input data formats / tokenizers / segmenters / semantic models / re-ranking methods / fulltext search engines / ...

Similarity Search: Cosine / euclidean / ... similarity.

of Semantic Vectors: LSI / deep learning / doc2vec / ...

using Fulltext Search Engines: Sphinx, Lucene, Elasticsearch, Solr, ...
Questions?
Illustrations by Jiří Franek.

