Seminar 1. Embeddings, DSSM

Nikolay Karpachev
5.02.2024

Word embeddings. Evaluation

Q: How to evaluate embeddings (metric space) quality?

Word embeddings. Evaluation

Q: How to evaluate embeddings (metric space) quality?

Intrinsic quality measures

- Some specific intermediate task
- Fast to compute
- Should capture embeddings structure
- Need to ensure correlation with real-life tasks

Extrinsic quality measures

- Evaluation on actual downstream task
- Slow to compute
- Reliable

Intrinsic Evaluation

- Word analogies

a : b : : c : ?

$$d = \underset{i}{\operatorname{argmax}} \frac{(x_b - x_a + x_c)^T x_i}{\|x_b - x_a + x_c\|}$$

max cosine similarity

Input	Result Produced
Chicago : Illinois : : Houston	Texas
Chicago : Illinois : : Philadelphia	Pennsylvania
Chicago : Illinois : : Phoenix	Arizona
Chicago : Illinois : : Dallas	Texas
Chicago : Illinois : : Jacksonville	Florida
Chicago : Illinois : : Indianapolis	Indiana
Chicago : Illinois : : Austin	Texas
Chicago : Illinois : : Detroit	Michigan
Chicago : Illinois : : Memphis	Tennessee
Chicago : Illinois : : Boston	Massachusetts

Possible metrics

- accuracy over corpus
- mean position of correct answer in top-K proximal

Example sources: https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes02-wordvecs2.pdf

Intrinsic Evaluation

- Synonym banks / thesauri
- Correlation with similarity judgements from humans
- Clusterization quality
- ...

DSSM

DSSM (Deep Structured Semantic Model)

- shared metric space for objects
- not necessarily homogeneous
- e.g. text + images, text + audio

Any ideas?

DSSM

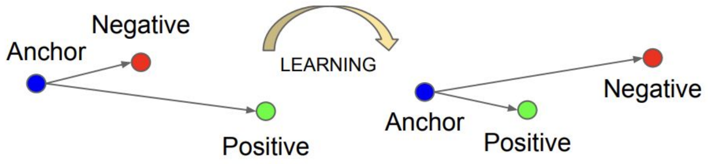
DSSM (Deep Structured Semantic Model)

Textual data	→	Text encoder (word2vec, RNN etc.)	*A (anchor)*
Corresponding image	→	Image encoder (CNN etc.)	*P (positive)*
Non-correspon ding image	→	Image encoder (CNN etc.)	*N (negative)*

DSSM

DSSM (Deep Structured Semantic Model)

Textual data	→	Text encoder (word2vec, RNN etc.)	*A (anchor)*
Corresponding image	→	Image encoder (CNN etc.)	*P (positive)*
Non-correspon ding image	→	Image encoder (CNN etc.)	*N (negative)*



Triplet loss (max margin loss)

Goal: $\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$

Loss: $\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$

DSSM

DSSM Applications

- Cross-domain search (image by text)
- Intra-domain search (just like word2vec, metric space image2image, doc2doc etc.)

DSSM

DSSM Applications

- Cross-domain search (image by text)
- Intra-domain search (just like word2vec, metric space image2image, doc2doc etc.)

- Transfer knowledge from resource-rich domain to resource-poor

Semantic Search

- Embedding space (DSSM)
- **Goal**: retrieve most similar values to a given query

Q: How to retrieve most similar values by query?

Semantic Search

- Embedding space (DSSM)
- **Goal**: retrieve most similar values to a given query

Q: How to retrieve most similar values by query?
- Get embedding via corresponding encoder
- Cosine similarity
- **Too slow**

Approximate Semantic Search

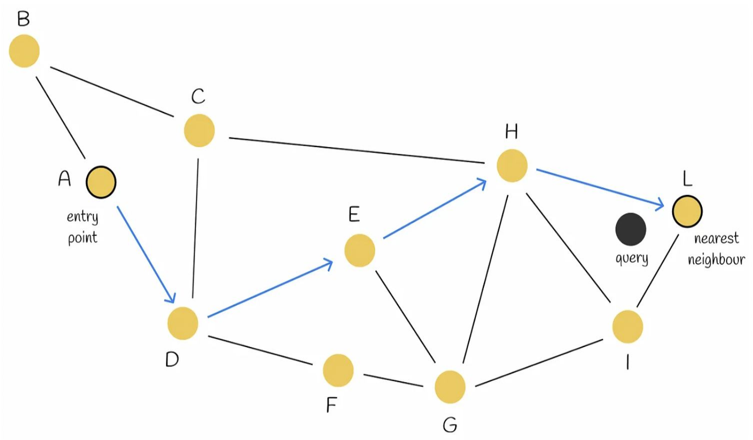
- **HNSW / NSW ([Hierarchical] Navigable Small World)** - approximate nearest neighbour search

1. Construct a graph in embeddings space [vertices == values]
2. Embed query, traverse graph of values by comparing current node proximity to query embedding

NSW

Search

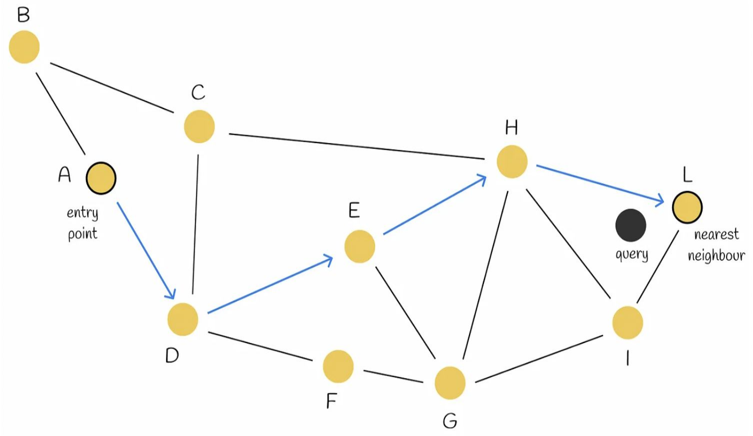
- Pick initial graph node N
- Compute *d(Q, Adj)* for every neighbour Adj
- *Adj_closest:= {A: d(Q, A) <= d(Q, Adj) for every Adj}*
- Repeat until *d(Q, N) < d(Q, Adj_closest)*



NSW

Search

- Pick initial graph node N
- Compute *d(Q, Adj)* for every neighbour Adj
- *Adj_closest:= {A: d(Q, A) <= d(Q, Adj) for every Adj}*
- Repeat until *d(Q, N) < d(Q, Adj_closest)*

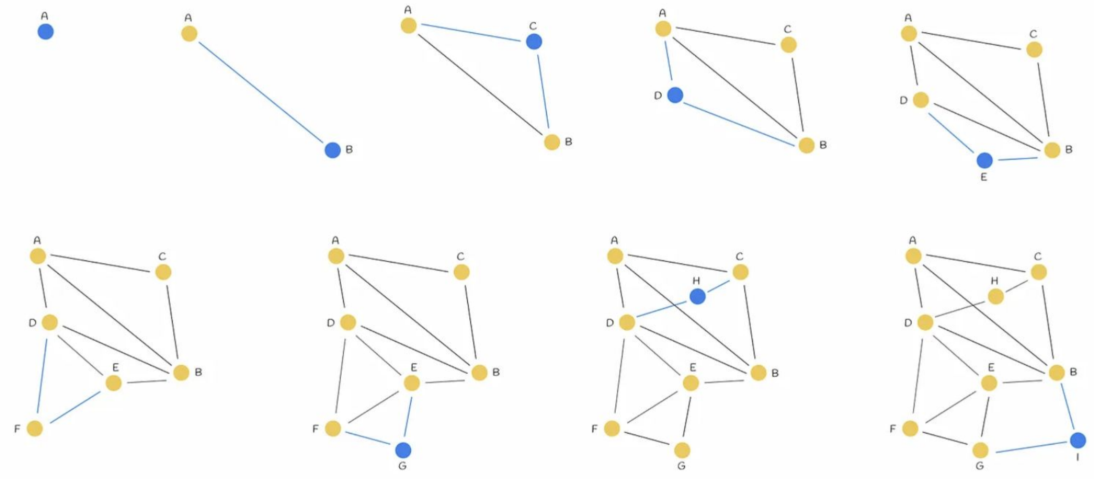


Expected closest path length: O(log n)

NSW

Graph construction

- Shuffle dataset points
- Sequentially insert
  - Link to M closest in current graph



NSW graph construction with M = 2

HNSW

Goal: Avoid dense clusters

<u>Construction</u>

- Build NSW graph
- With probability *p*, add node to next NSW layer

<u>Search</u>

- Start with top layer (sparse!)
- Find current layer optimum, traverse ancestor layer from retrieved node