IN3120 week 12

Group 1

Agenda

- 2023 exam walkthrough
  - 4 tasks, 14 sub-tasks
- Shoutout

Task 1 – Vector spaces

a) Sparse vs. dense vector space
b) Cosine similarity
c) ANN indexes
d) List 5 types of ANN indexes

Sparse vs. dense vector space (a)

- Two ways to think about vector space for text
  - (A) Sparse high dimensional – dimension for each term
  - (B) Dense low dimensional/embedding space – few hundred dimensions

(i) Briefly discuss how a document could be placed in vector space (A). Outline pros and cons for this representation

(ii) Briefly and on a high level, what are the general ideas how a word is placed in this embedding space? How to place a document in this space? Outline pros and cons of representation (B)

1 (a) (i)

- We can use TF-IDF
- Boosts dimensions for terms often occurring in this document, and rarely in other documents. Value is 0 for terms not occurring in the document
- Pro: Can often use linear techniques when classifying, since it often is easy to find a separating hyperplane
- Con: Superficial term differences (synonyms etc.)

1 (a) (ii)

- A word is known by the company it keeps
  - Words often semantically related
  - Capture context by considering co-occurrence
- Can compute embeddings with a small neural network trying to predict a blanked-out word in a sequence, and read values from internal nodes to produce the embedding. Embeddings for words can be used as building blocks for longer text
- Pro: Distances in the embedding space correlate with semantics. Can use this for simple reasoning
- Con: Doesn’t preserve the contents of the document as well as sparse

Sparse vs. dense vector space (b)

- We have the vectors [0.6, 0.2, 0.8] and [1.0, 0.1, 0.9]. Show how to compute the cosine similarity

1 (b)

- Cosine similarity steps:
  1. Calculate Dot product of the vectors
  2. Calculate L_2-norm of the vectors
  3. Divide dot product by the product of L_2-norm

$$Cosine(x, y) = \frac{dot_{product(x,y)}}{L_2norm(x) * L_2norm(y)}$$

$$x = [0.6, 0.2, 0.8]$$
$$y = [1.0, 0.1, 0.9]$$

- Calcluations
  1. $dot\ product(x, y) = (0.6 * 1.0) + (0.2 * 0.1) + (0.8 * 0.9) = 1.34$
  2. $L_2(x) = \sqrt{0.6^2 + 0.2^2 + 0.8^2} = \sqrt{1.04}$
  3. $L_2(y) = \sqrt{1.0^2 + 0.1^2 + 0.9^2} = \sqrt{1.82}$
  4. $cosine(x, y) = \frac{1.34}{\sqrt{1.04} * \sqrt{1.82}} = 0.974$

Sparse vs. dense vector space (c)

- Explain what an ANN index is, and why it is useful

1 (c)

- Closeness of vectors in an embedding space means «semantic similarity»
- If we want to query this space, the docs close to the query vector are similar
- An ANN index is an engine that allows for kNN lookups (finding the k closest documents), but scales better with larger dimensionality and number of vectors, because we sacrifice the exactness of kNN for speed
- We aren’t guaranteed to find the exact k nearest neighbours, but we will find close nodes quickly

Sparse vs. dense vector space (d)

- List at least 5 ANN index strategies, and explain the thinking behind each strategy
- https://github.com/aohrn/in3120-2024/blob/main/slides/approximate-nearest-neighbours.pdf

1 (d)

(i) **Brute force**: Scan through all vectors and assess their distance to the query vector (only works up to a certain point. Can be an option to split the search area with other algorithms first

(ii) **Tree-based**: Recursively build a tree by partitioning the embedding space. Leaf nodes are partitions. Only query the elements in the leaf node of the query (ANNOY)

(iii) **Locality-sensitive hashing**: Use hash functions to bucket elements together. If elements are in the same bucket, they are probably similar. Run query through the same hash functions and calculate distance to elements in the buckets of the query

1 (d)

(iv) **Clustering-based (quantization)**: Cluster the vectors to reduce the size of the dataset. Replaces vectors with leaner, approximate representations. Use this reduced dataset to query

(v) **Graph-based**: Create a layered structure of graphs. Each layer gets more detailed. Nodes are vectors and edges relate vectors close to each other. Intuitively: «skip-pointer graphs»

Task 2 – Measuring relevance

a) F_β-score
b) Precision-recall curve
c) MAP
d) Kandall’s tau distance

Measuring relevance (a)

- Describe F_β-score and define it in terms of precision and recall. What does β control? What is the F_1-score if P=0.1 and R=0.5

2 (a)

- F_β-score is the weighted harmonic mean of precision and recall
- Great P and R is ideal, but we might have to compromise. F is a measure that collapses both into a single number, and is therefore often convenient
- β controls how much we emphasize precision vs. recall. β < 1 emphasizes precision, β > 1 emphasizes recall
- P=0.1 and R=0.5

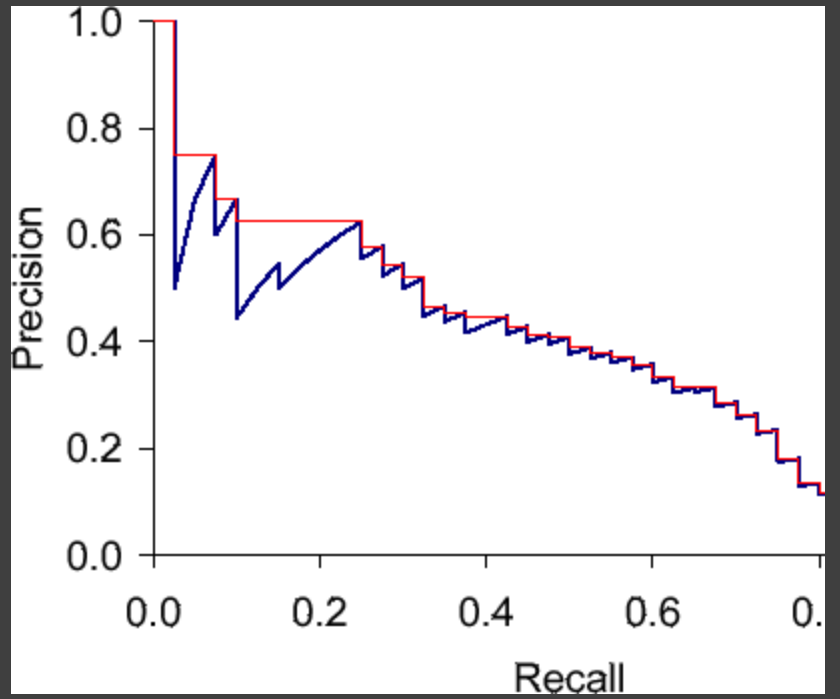
$$F_{\beta=1} = \frac{2PR}{P+R} = \frac{(2 * 0.1 * 0.5)}{0.1 + 0.5} = \frac{1}{6} = 0.167$$

Measuring relevance (b)

- What is a precision-recall curve and how do we generate it? What is an interpolated precision-recall curve?

2 (b)

- If we consider top k elements in a ranked result set. For each choice of k, we can calculate precision and recall. This is the precision-recall curve. This curve shows how precision often falls with increasing recall
- Interpolated PR-curve is when we set precision as the highest P to the right of the current position in the curve (the red line)



Measuring relevance (c)

- What is the MAP (Mean Average Precision) score of the results:
  - RRNRNNNR
  - RNRR

2 (c)

- Average_precision(RRNRNNNR)

$$\frac{\frac{1}{1}+\frac{2}{2}+\frac{3}{4}++\frac{4}{8}}{4}=0.8125$$

- Average_precision(RNRR)

$$\frac{\frac{1}{1}+\frac{2}{3}+\frac{3}{4}}{3}=\frac{29}{36}=0.8055$$

- MAP

$$\frac{0.8215+0.8055}{2}=0.8090$$

Measuring relevance (d)

- Describe how Kendall’s tau is calculated on a high level, and what is Kendall’s tau between
  - L = [A, C, B, D]
  - P = {(A, B), (A, C), (A, D), (B, C), (B, D), (C, D)}

2 (d)

- A pair in P will either be in agreement or disagreement with the order of L. Given all pairs in P, we calculate KT by counting all agreements (X) and disagreements (Y), and consider their difference (X-Y) as a fraction of the total number of pairs in P (X+Y). This gives a value in the range [-1, 1] where the extremes represent perfect agreement/disagreement
  - KT=(X-Y)/(X+Y)
- For L = [A, C, B, D], P = {(A, B), (A, C), (A, D), (B, C), (B, D), (C, D)}, we have 5 agreements and 1 disagreement:
  - KT = (5-1)/(5+1)=4/6=0.667

Task 3 – Mixed grill with carrots

a) Multinomial Naive Bayes
b) Postings list & inverted index
c) Gap encoding, VB-encoding elias-gamma encoding
d) Bloom filter

Mixed grill with carrots (a)

Based on the training data, how would a mulinomial naive Bayes classifier classify an unseen document *d* = «*carrot carrot carrot toffee jellybean*»?

Show how to estimate all required priors and conditionals. Show how to combine these to arrive at *Pr(healthy | d)* and *Pr(unhealthy | d)* respectively. Use add-one smoothing when calculating conditionals

document_id	body	class
1	*carrot broccoli carrot*	*healthy*
2	*spinach carrot carrot*	*healthy*
3	*carrot mango*	*healthy*
4	*toffee jellybean carrot*	*unhealthy*

3 (a)

- Priors
  - Pr(healthy) = ¾
  - Pr(unhealthy) = ¼
- Smoothed conditionals
  - Pr(carrot | healthy) = (5+1)/(8+6) = 3/7
  - Pr(toffee | healthy) = (0+1)/(8+6) = 1/14
  - Pr(jellybean | healthy) = (0+1)/(8+6) = 1/14
  - Pr(carrot | unhealthy) = (1+1)/(3+6) = 2/9
  - Pr(toffee | unhealthy) = (1+1)/(3+6) = 2/9
  - Pr(jellybean | unhealthy) = (1+1)/(3+6) = 2/9

3 (a)

- d = *carrot carrot carrot toffee jellybean*
- $\Pr(\text{healthy} \mid d) = 3/4 * (3/7)^3 * 1/14 * 1/14 \approx 0.0003$
- $\Pr(\text{unhealthy} \mid d) = 1/4 * (2/9)^3 * 2/9 * 2/9 \approx 0.0001$

Mixed grill with carrots (b)

- Imagine an inverted index made from the training data in task (a). Assume each posting contains (docId, term_frequency). List all posting lists

3 (b)

- 6 unique words → 6 posting lists
  - carrot → `[{docId: 1, tf: 2}, {docId: 2, tf: 2}, {docId: 3, tf: 1}, {docId: 4, tf: 1}]`
  - broccoli → `[{docId: 1, tf: 1}]`
  - spinach → `[{docId: 2, tf: 1}]`
  - mango → `[{docId: 3, tf: 1}]`
  - toffee → `[{docId: 4, tf: 1}]`
  - jellybean → `[{docId: 4, tf: 1}]`

document_id	body	class
1	*carrot broccoli carrot*	*healthy*
2	*spinach carrot carrot*	*healthy*
3	*carrot mango*	*healthy*
4	*toffee jellybean carrot*	*unhealthy*

Mixed grill with carrots (c)

- Consider the same inverted index as (b)
  - (i) How many bytes are necessary to compress the *carrot* posting list, when combining simple gap-encoding with VB-encoding? Explain your reasoning
  - (ii) How many bits would you need to store the compressed posting list for *carrot*, when combining simple gap-encoding with Elias gamma-encoding? Explain your reasoning

3 (c) (i)

- There are 8 integers in the posting list
- No matter if we store gaps or docId, all values are below 128, so we only need 1 byte per int
- 8 bytes in total

3 (c) (ii)

- When gap-encoding the posting list for carrot, it will look like

```
[{gap: 1, tf: 2}, {gap: 1, tf: 2}, {gap: 1, tf: 1},
{gap: 1, tf: 1}]
```

- 1 can be encoded with 1 bit
- 2 can be encoded with 3 bits
- 6 * 1 + 2 * 3 = 12
- Gap encoding: length of offset in unary + offset of binary number

Table 5.5: Some examples of unary and γ codes. Unary codes are only shown for the smaller numbers. Commas in γ codes are for readability only and are not part of the actual codes.

number	unary code	length	offset	γ code
0	0			
1	10	0		0
2	110	10	0	10,0
3	1110	10	1	10,1
4	11110	110	00	110,00
9	1111111110	1110	001	1110,001
13		1110	101	1110,101
24		11110	1000	11110,1000
511		111111110	11111111	111111110,11111111
1025		1111111110	0000000001	1111111110,0000000001

Mixed grill with carrots (d)

- Given a 16 bit Bloom filter, 3 hash functions and the hash values in the table
  - (i) What does the filter look like before inserting anything, after inserting *carrot* and after inserting *carrot* and *toffee*
  - (ii) Given *carrot* and *toffee* have been inserted, what will the filter answer when querying *steak* and *carrot*? Why?
  - (iii) How can the Bloom filter be modified to reduce the probability of false positives?

Hash function h	Value x	Hash value $h(x)$
h_1	*carrot*	12
h_1	*toffee*	0
h_1	*steak*	7
h_2	*carrot*	7
h_2	*toffee*	12
h_2	*steak*	15
h_3	*carrot*	15
h_3	*toffee*	3
h_3	*steak*	11

3 (d) (i)

- Before inserting anything: []

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- After inserting *carrot*: [7, 12, 15]

0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1

- After inserting *carrot* and *toffee*: [0, 3, 7, 12, 15]

1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1

3 (d) (ii)

- Querying *carrot* returns true/«member» because 7, 12, 15 are all true/set
- Querying *steak* returns false/«not member» because 11 is false/not set

3 (d) (iii)

- To reduce false positives, we can
  - add more storage bits or
  - change the number of hash functions

Task 4 – Aproxximat matching

a) Finding similar terms in a trie
b) Edit distance with phonetic hashing

Aproxximat matching (a)

- Given a trie encoding a string collection D and a query q, we want to find all strings in D, within k edits from q. We can assume k is a small number

4 (a)

- Same as oblig b-2 (can read both paper and code on Github)
- Perform recursive dfs search on trie
  - Whenever we traverse another layer in the trie, we check the distance between the current string and the query using Damerau-Levenshtein distance
- If > k edits away or leaf node is reached, we stop
- Whenever we find a match (final node within k edits), send to sieve and store for later

Aproxximat matching (b)

- Propose a way to alter the (a) algorithm to combine edit distance with phonetic hashing. The algorithm should find all strings in D where the phonetic hashes between a string in D and q differ at least k edits

4 (b)

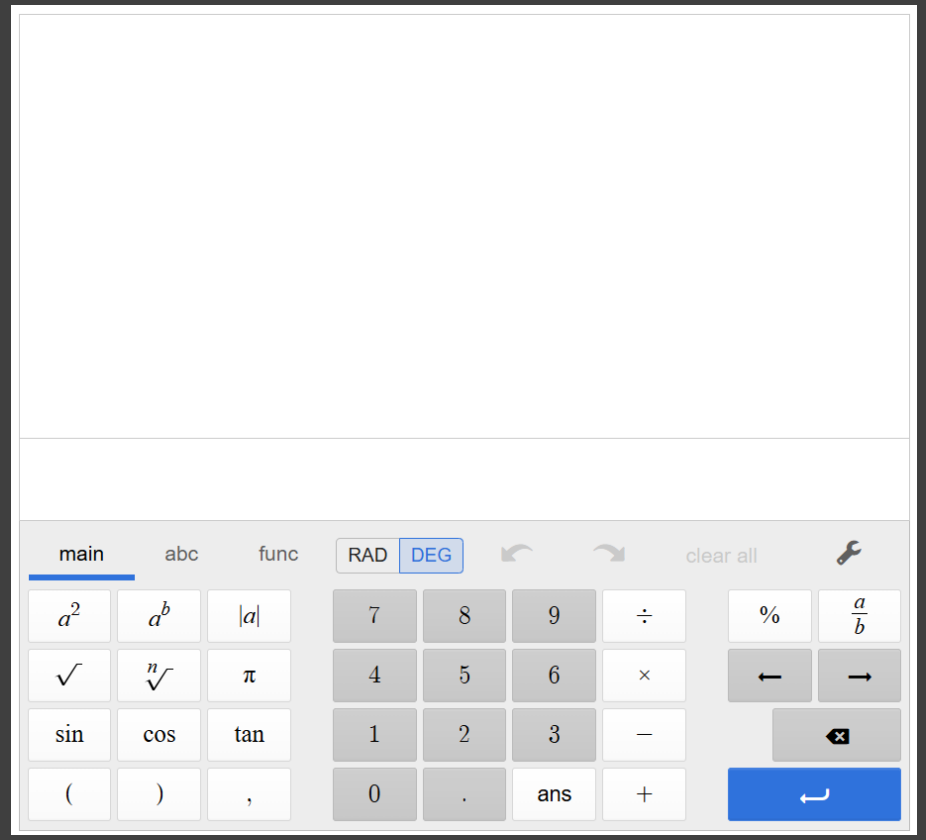
- Use the same algorithm as (a), but extend trie to hold metadata along with each final state, and some pre- and post-processing
- When making the trie: Instead of storing the original string, store the phonetic hash. Each final state holds metadata to trace backto the original string
- When querying: Compute the phonetic hash of q, and find matches within k edits using the algorithm from (a)

Agenda

- 2023 exam walkthrough
  - 4 tasks, 14 sub-tasks
- Shoutout

Shoutout of the week

- Desmos (Inspera-kalkulatoren?)
- https://www.desmos.com/scientific



15 min break