به نام خدا

دانشگاه تهران
دانشکدگان فنی
دانشکده مهندسی برق و کامپیوتر

درس بازیابی هوشمند اطلاعات

پاسخ بخش تئوری تمرین ۴

نام و نام خانوادگی: ارشیا مخلص

شماره دانشجویی:810104247

آذر ماه ۱۴۰۴

فهرست

مقدمه

اینجانب ارشیا مخلص با شماره دانشجویی 810104247 برای برقراری بهتر ارتباط با درس و اسلاید و همچنین خود زبان انگلیسی تمارین خود را به زبان انگلیسی تحویل میدهم.

تایید می‌کنم که از LLM ها مطابق با دستورالعمل‌های بارگذاری شده در سامانه Elearn درس به طور مسئولانه استفاده کرده‌ام. تمام اجزای کار خود را درک می‌کنم و آماده بحث شفاهی درباره آنها هستم.

با تشکر

ارشیا مخلص

Q1-ANS

PART A – PARADIGMATIC VS. SYNTAGMATIC

In word association relations there Are two main ways to say 2 words are semantically connected which are **Paradigmatic** and **Syntagmatic.**

- **Paradigmatic:** This relation happens between 2 words when they can substitute each other in a sentence.
- **Syntagmatic:** This relation happens between 2 words when they occur together a lot

PART B

I didn't really understand the question but I guess that Paradigmatic relation is about semantic similarity (when the meanings of 2 words are similar), and syntagmatic is about contextual similarity (2 words appear together a lot).

PART C

Paradigmatic:

EX: "The **cat** sat on the mat" and "The **dog** sat on the mat"

Cat and dog have Paradigmatic relation which allows them to substitute each other in a sentence.

Syntagmatic:

EX: "I used a **knife** to **cut** an apple"

Cut and Knife have a Syntagmatic relation because usually most of the times we want to use a knife we want to cut something.

Q2-ANS

PART A

MI or mutual information is the same as syntagmatic relation and contextual similarity. MI is about the chance that two words appear together a lot, if the chance that two words appear together is higher than average it means that the MI measure is high for them. Be aware that MI isn't about semantic similarity.
Based on what MI stands for if MI measure is high for 2 words it means those words appear more than average together and this means that the possibility that these words are in the same subject increases.

PART B

Like question 1 part c **knife** and **cut** have high MI but they have no sematic similarity.

Q3-ANS

PART A

In Word Embedding we store 2 types of information:

1- Semantic Info:

Semantic Information defines how much two words have the same meaning

EX: *intelligence & smart*

This means that intelligence and smart vectors are close together in Word Embedding vectors

2- Syntactic Info:

Syntactic Information is about how much two words are grammatically similar and store patterns of words

EX: *walked & played*

This means that walked and played are closer together than walked and play because they have a **past tense** direction. Also, we can conclude that:

$$\boldsymbol{walked - walk = played - play}$$

PART B

Linearity in Word Embedding means that moving from one word vector to another by adding a relation vector corresponds to a meaningful semantic transformation, such as **gender shift** in $\boldsymbol{king \rightarrow queen}$ transformation.

Relations like $king - man + woman \approx queen$ describe gender shift and how masculine energy shifts into feminine energy $\overrightarrow{v_{woman}} - \overrightarrow{v_{man}} \approx \overrightarrow{v_{gender}}$ and in the same time it is holding the meaning of a **royal** person so we are shifting a word from ”masculine + royal” → “feminine + royal”
In general, we can see semantic transformation like gender shift, verb tense, plurality and … .

Q4-ANS

Word2Vec uses CBOW and Skip-gram to train the model and predict vector of words.

CBOW

In Continuous Bag Of Words the model tries to predict the missing word inside a sentence using other words of sentence.

EX: “the cat sat on the mat”

“The … sat on the mat” → using CBOW → “the cat sat on the mat”

Input: The vector of surrounding words
Output: The probability distribution of middle(missing) word. usually averages the input vectors.

SKIP-GRAM

In Skip-gram we try to train the model and predict the sentence using other words of the middle word.

EX: “the cat sat on the mat”

“… cat … … … ….” → using Skip-gram → ““the cat sat on the mat”

Input: the vector of the middle word.

Output: the probability distribution of the surrounding words.

WHICH METHOD FOR FREQUENT WORDS?

CBOW.

CBOW is better for frequent words. In CBOW method model trains itself using the whole sentence and it average lots of sentences That means that we are extracting data from lots of different sentences and extract context from each one of them but, because of the CBOW method for each sentence we try to learn the model with just one vector (the vector of the missing or middle word). So, it has lower computational weight than Skip-gram and extracts its context from different sentences.

WHICH METHOD FOR RARE WORDS?

Skip-gram.

Skip-gram is better for rare words. in Skip-word we train the model using many vectors for each sentence which means that we use every data that we can extract and train the model from each sentence and we are using what we have in the best way.

Q5-ANS

COMPARE APPROACHES & LABLES

Approach	Models	Label
Pointwise	Relevance score of a document for a query	Relevance Value (e.g. [0,10] or {0,1} or …) of a doc for a query
Pairwise	Preference between 2 documents for the same query	Preferred document (e.g. doc_A is better than doc_B
Listwise	Ranked document list of all documents for a query	Relevance vector ([docA , docC, docB, …])

LOSS FUNCTION FORM

- **Pointwise**
  - Typical losses:
    - Regression (e.g., mean squared error) when labels are real-valued relevance scores.
    - Classification loss (e.g., logistic loss) when labels are binary relevant/irrelevant.
  - Optimizes per-document prediction error independently.
- **Pairwise**
  - Typical losses:
    - Pairwise ranking loss, e.g., hinge loss or logistic loss on score differences $s(q, d_i) - s(q, d_j)$, encouraging d_i to score higher than d_j when d_i is more relevant.
  - Directly penalizes mis-ordered document pairs for the same query.
- **Listwise**
  - Typical losses:
    - Direct optimization of a list-level objective or its surrogate, e.g., ListNet, ListMLE, or LambdaMART that uses listwise gradients.

- Uses the whole ranked list and its relevance labels to define a loss that approximates rank-based metrics (like NDCG).

ADVANTAGES VS. LIMITATION

Approach	Advantages	Limitations
Pointwise	Simple formulation; easy to use off-the-shelf regression/classification models; scales well.	Ignores relative ordering; not aligned with ranking metrics; may focus on absolute scores instead of rank positions.
Pairwise	Directly models ordering; more aligned with ranking than pointwise; usually better ranking quality.	Still does not fully consider the whole list; number of pairs can grow quadratically; optimization is not directly on NDCG/MAP.
Listwise	Most faithful to ranking goal; considers whole list structure; can better optimize top-ranked positions.	More complex objectives; training can be heavier; requires careful implementation and often more data.

COMPATIBILITY WITH NDCG

- **Pointwise**
  - Compatibility: Weak.
  - Reason: Minimizes per-document error and does not know that errors at the top of the list should be penalized more than at the bottom, while NDCG is highly position-sensitive.
- **Pairwise**
  - Compatibility: Moderate.
  - Reason: Corrects pairwise orderings, which improves metrics like NDCG and MAP, but all pairs are usually treated similarly unless specially weighted by position.
- **Listwise**
  - Compatibility: Strongest.
  - Reason: Many listwise methods (e.g., LambdaRank/LambdaMART) define gradients proportional to changes in NDCG when swapping

documents; thus they are explicitly designed to improve NDCG or similar list-based metrics.

Q6-ANS

Λ-GRADIENT IN LAMBDARANK

- LambdaRank is a listwise learning-to-rank method that defines the **gradient of the loss** for each document pair indirectly, through quantities called **λ-gradients**.
- For a given query, consider a pair of documents d_i and d_j with different relevance labels.
    - The model produces scores s_i and s_j.
    - LambdaRank defines a **λ-value** (λ-gradient) for each document as:
        - A function of the **score difference** $s_i - s_j$ (as in pairwise logistic loss).
        - **Multiplied** by the **absolute change in the ranking metric** (e.g., | ΔNDCG |) if these two documents were swapped.
- These λ-values are then used as the effective gradients for backpropagation in the underlying model (e.g., Gradient Boosted Trees).

WHY SWAPS AT THE TOP OF THE LIST GET LARGER GRADIENTS

- Metrics like **NDCG** discount lower ranks: errors at rank 1 or 2 affect the score much more than errors at rank 50.
- When two documents are swapped near the **top** of the ranked list:
    - The change in NDCG, | ΔNDCG |, is **large**.
    - Therefore, the λ-gradient magnitudes for those documents become **large**, leading to stronger parameter updates.
- When the swap happens near the **bottom**:
    - | ΔNDCG | is very **small** because the discount factor for those positions is small.
    - The λ-gradients are correspondingly small, so the model does not spend much capacity correcting low-impact errors.

Thus, LambdaRank **naturally focuses learning on the top positions**, where ranking quality matters most for NDCG.

HOW LAMBDARANK ALIGNS WITH NDCG AND OTHER RANKING METRICS

- The λ-gradient for each pair is explicitly scaled by the **change in the ranking metric** (e.g., NDCG) that would result from correcting that pair's order.
- As a result:
    - The optimization process **implicitly maximizes NDCG** (or the chosen metric) instead of an abstract surrogate loss disconnected from evaluation.
    - The model is **metric-aware**: it learns to prioritize changes that give the largest improvement in the target ranking metric.
- This makes LambdaRank (and its tree-based extension LambdaMART) one of the most **compatible** methods with position-sensitive metrics like **NDCG** and **MAP**, especially compared to pure pointwise or vanilla pairwise methods.

Q7-ANS

A) FEATURE VECTORS FOR EACH WORD

interpretable feature vectors are:

- dog = $[1, 1, 0, 0, 0, 0]$
- cat = $[1, 1, 0, 0, 0, 0]$
- bird = $[1, 0, 1, 0, 0, 0]$
- apple = $[0, 0, 0, 1, 0, 1]$
- banana = $[0, 0, 0, 1, 0, 1]$
- car = $[0, 0, 0, 0, 1, 0]$

Order of dimensions:

[IS_ANIMAL, HAS_TAIL, CAN_FLY, IS_FRUIT, IS_VEHICLE, IS_FOOD].

B) COSINE SIMILARITIES

Cosine similarity:

$\cos(x, y) = \frac{x \cdot y}{\|x\|\|y\|}$(dot product over product of vector norms).

- dog vs cat:
  - dot = $1 \cdot 1 + 1 \cdot 1 = 2$
  - $\| dog \| = \| cat \| = \sqrt{1^2 + 1^2} = \sqrt{2}$
  - $\cos(dog, cat) = \frac{2}{\sqrt{2}\sqrt{2}} = 1$
- dog vs bird:
  - dot = $1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 = 1$
  - $\| dog \| = \sqrt{2}, \| bird \| = \sqrt{1^2 + 1^2} = \sqrt{2}$
  - $\cos(dog, bird) = \frac{1}{2} = 0.5$
- apple vs banana:
  - dot = $1 \cdot 1 + 1 \cdot 1 = 2$(from is_fruit and is_food)
  - $\| apple \| = \| banana \| = \sqrt{1^2 + 1^2} = \sqrt{2}$
  - $\cos(apple, banana) = 1$
- dog vs apple:
  - dot = 0(no shared 1s)
  - $\cos(dog, apple) = 0$

- apple vs car:
  - dot = 0(no shared 1s)
  - cos $(apple, car) = 0$

C) ANALYSIS OF RESULTS

MOST SIMILAR PAIRS

  - dog–cat: similarity = 1
  - apple–banana: similarity = 1
  - Next is dog–bird: 0.5
  - Others: 0 (dog–apple, apple–car).

This matches intuitive category structure:

  - dog, cat, bird share animal–related features;
  - apple and banana share fruit/food features;
  - car is clearly separated as a vehicle.

WHAT KIND OF SIMILARITY IS BEING MEASURED?

- These interpretable vectors measure shared explicit semantic properties (hand-designed features like “is_animal”, “is_food”).
- The similarity here is feature-based taxonomic similarity: words are similar if they share many human-defined attributes (same category, same role).

HOW THIS DIFFERS FROM WORD EMBEDDING SIMILARITY

- Word embeddings (e.g., Word2Vec, GloVe) encode similarity based on distributional context (words that appear in similar contexts have similar vectors), not on explicit human-defined features.
- Embeddings can capture:
  - Topical/relational similarity (doctor–hospital, apple–tree) even if features differ.
  - Nuanced relations (analogy directions) that are not directly visible as simple binary attributes.
- In contrast, this interpretable space:
  - Cannot capture all nuanced relations.
  - But gives transparent, explainable reasons for similarity (“both are fruits and foods”).

WHICH REPRESENTATION SUITS PARADIGMATIC VS SYNTAGMATIC?

- Paradigmatic similarity (substitutability, same role/category):
    - Better modeled by interpretable feature vectors, since they explicitly encode category-like properties (animal, fruit, vehicle).
- Syntagmatic similarity (co-occurrence in context, collocations):
    - Better modeled by distributional embeddings, since they arise from words that appear together or in similar contexts (e.g., “doctor–patient”, “car–road”, “apple–pie”).

So, this interpretable feature space is more naturally aligned with paradigmatic relations, while standard word embeddings are powerful at capturing syntagmatic associations as well as more subtle semantic patterns.