DAT640 2024 autumn, Final Exam - Answer Key

Note: all auto-graded answers are double-checked manually and corrected for rounding errors.

General approach to grading the essay questions
Even if a solution contains the answer that should give a certain score based on the provided solution key, points may be deducted if the answer has a mix of correct and incorrect statements, is too long, or has largely irrelevant content w.r.t. the question.

2 Text representation (2 points; -1 if incorrect)

Using a sparse vector representation, each document is represented by a vector of length n, where n is?

- ( ) Total number of document terms (i.e., document length)
- ( ) Number of distinct terms in the document
- (X) Number of distinct terms in the whole collection

3 Indexing (3 points; -1 if incorrect)

DocID: 1501 Snooker is a cue sport that was first played by UK Army officers stationed in India.	DocID: 1502 The World Snooker Championship was a professional snooker tournament held in Sheffield.
DocID: 1503 Steve Davis OBE is an English retired professional snooker player from Plumstead.	DocID: 1504 Ben Nevis is the highest mountain in Scotland and UK.

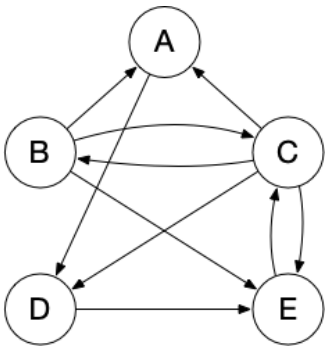
Above are excerpts from four Wikipedia articles. We want to create a separate index for anchor text.
How many posting lists would be in that index? Assume that text processing is done by lowercasing text and removing punctuation; no stemming or stopword removal are applied.

- ( ) 9
- ( ) 10
- (X) 11
- ( ) 13

4 Term weighting (2 points)

In the vector space model, which of the two term weighting heuristics is ineffective if a query has a single term? Explain briefly why.

Answer: IDF (1p); it's the same for all documents and hence does not affect the ranking (1p).

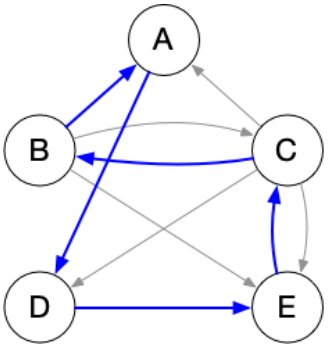


5 PageRank (3 points)

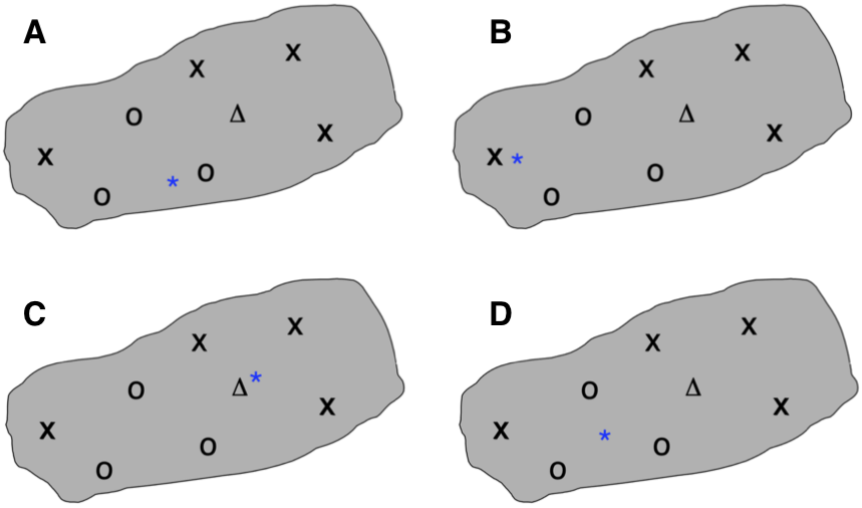
How many edges may be removed from this link graph without resulting in any rank sinks, while ensuring that the graph remains connected (i.e., there is a path between any two nodes)?

Answer: 5

To ensure that there are no rank sinks, each node must have at least one outgoing edge. We can fulfill this requirement minimally (i.e., having exactly one outgoing edge for each node) while keeping the graph connected. For example:



6 Relevance feedback (3 points; -1 if incorrect)



$$\vec{q}_m = \alpha\vec{q} + \frac{\beta}{|D^+|}\sum_{d\in D^+}\vec{d} - \frac{\gamma}{|D^-|}\sum_{d\in D^-}\vec{d}$$

($\vec{q}$: original query vector; D^+, D^-: set of relevant and non-relevant feedback documents, respectively)

Consider applying the Rocchio algorithm with alpha=0.1, beta=0.9, and gamma = 0.
Which of the figures above show the position of the reformulated query?

In the figures, we represent query and document vectors in a 2-dimensional space, where the triangle refers to the original query, * refers to the reformulated query, O indicates relevant documents, and X indicates to non-relevant documents.

Answer: D

7 Retrieval evaluation (2 points; -1 if incorrect)

Which of the following is the most commonly used statistical test to evaluate the significance of retrieval results when comparing the performance of two retrieval systems?

- ( ) Pearson's correlation coefficient
- ( ) Chi-square test
- (X) Paired t-test
- ( ) Linear regression coefficient

8 Retrieval evaluation (5 points)

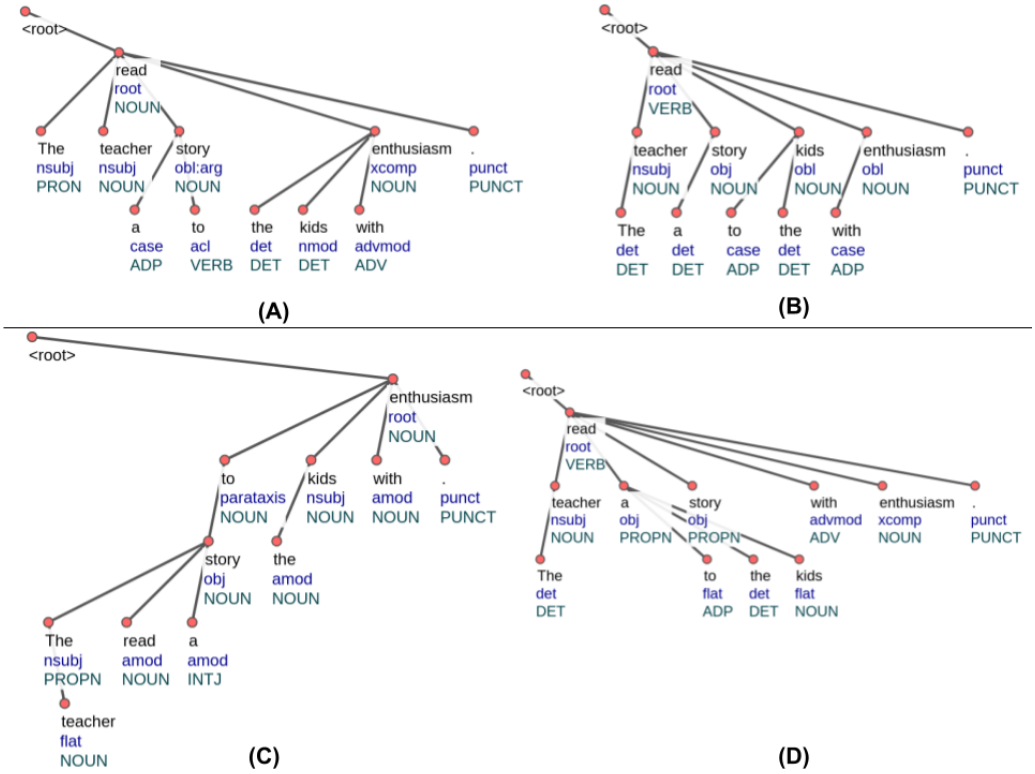
Below is the output of an IR system that returns a list of relevant (R) and non-relevant (N) documents for a query:

```
R N N R N R N N N N R N
```

The documents in this list are ordered from left to right, where the left most document is the top-ranked document. Assuming that there is a total of 6 known relevant documents for this query, answer the questions:

- What is Precision@10? [0.3] (3/10)
- What is Recall? [0.667] (4/6)
- What is the Average Precision (AP) of this system for this query? [0.394] (1/6 · (1+ 2/4+ 3/6 + 4/11))
- Assume that the system returned all 100 documents in the collection and these are the top-12 results. What would be the minimum AP? [0.412] (1/6 · (1+ 2/4+ 3/6 + 4/11 + 5/99 + 6/100))
- Assume that the system returned all 100 documents in the collection and these are the top-12 results. What would be the maximum AP? [0.529] (1/6 · (1+ 2/4+ 3/6 + 4/11 + 5/13 + 6/14))

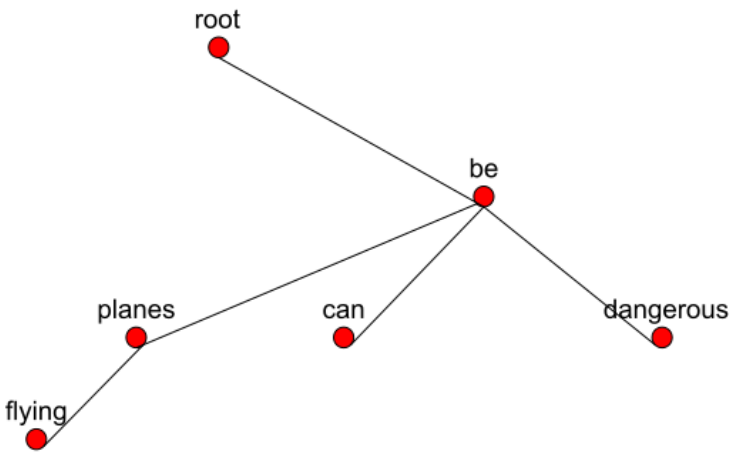
9 Parsing (3 points; -1 if incorrect)



Which of the above dependency parse trees represents the sentence "The teacher read a story to the kids with enthusiasm."

Answer: B

10 Tagging (5 points)



Let's have a sentence: 'Flying planes can be dangerous.' The sentence can have two different meanings. If the sentence has a meaning defined by the above dependency parse tree, what would be the POS tag of the word 'flying'? Use Penn Treebank tagset.

Scoring:

- Correct answer for full points (5p): JJ
- Incorrect answers for partial points: NN*,ADP (3p), JJR, JJs: (2p)

11 Coding (3 points)

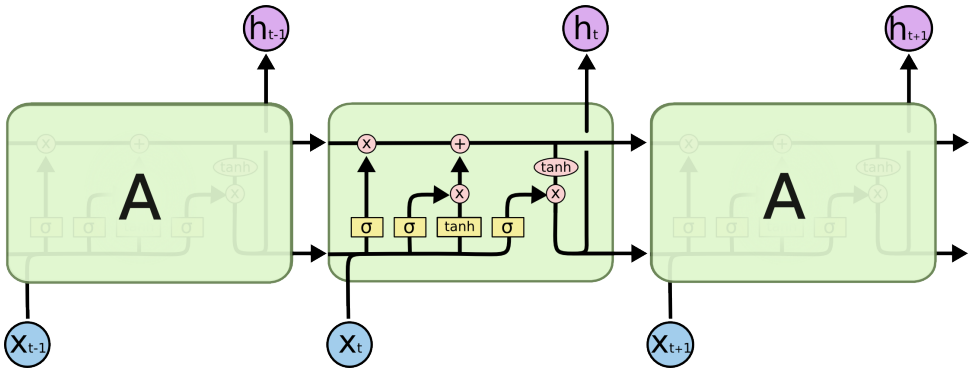
```python
import re
from collections import defaultdict, Counter


class Prediction:
    def __init__(self):
        self._bigram_counts = defaultdict(Counter)  # Holds counts of bigrams
        self._unigram_counts = Counter()  # Holds counts of individual words

    def preprocess(self, text):
        # Basic text preprocessing: Lowercase, remove non-alphanumeric characters
        text = text.lower()
        text = re.sub(r"[^a-zA-Z\s]", "", text)
        words = text.split()
        return words

    def train(self, corpus):
        words = self.preprocess(corpus)
        self._unigram_counts.update(words)
        bigrams = zip(words[:], words[1:])
        for w1, w2 in bigrams:
            self._bigram_counts[w1][w2] += 1

    def predict_next(self, word):
        # Get the most probable word following the given word
        following_words = self._bigram_counts[word]
        if not following_words:
            return []

        # Calculate probabilities
        total_count = sum(following_words.values())
        probabilities = {w: count / total_count for w, count in following_words.items()}

        # Sort by probability and return top prediction
        sorted_predictions = sorted(
            probabilities.items(), key=lambda item: item[1], reverse=True
        )
        return sorted_predictions[:1]


# Example usage:
corpus = "I am learning natural language processing. Language processing is very interesting."
model = Prediction()
model.train(corpus)

# Predict the next word after "language"
predictions = model.predict_next("language")
print("Predictions:", predictions)
```

What does the above code do?

Scoring: 3p for listing all correct answers, -1p for each incorrect answer (minimum score: 0).

- Correct answers (1p each):
  - It is a language model (or next word prediction).
  - It is using bigrams.
- Incorrect answers (-1p each):
  - Smoothing is implemented.
  - It is using trigrams.
  - It is using neural networks.
  - It is using Naive Bayes.

12 Neural architectures (2 points)



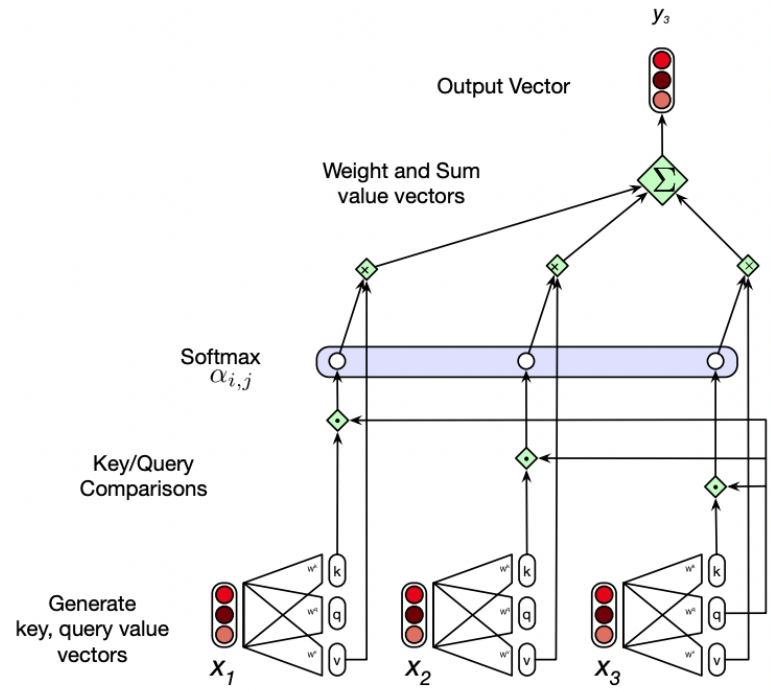
What is displayed in the above image?

- [X] LSTM architecture
- [X] RNN architecture
- [ ] CNN architecture
- [ ] Transformer architecture

Scoring:

- 2p if all correct
- 1p for each correct answer, -1p for each incorrect answer; minimum 0p

13 Large Language Models (5 points)

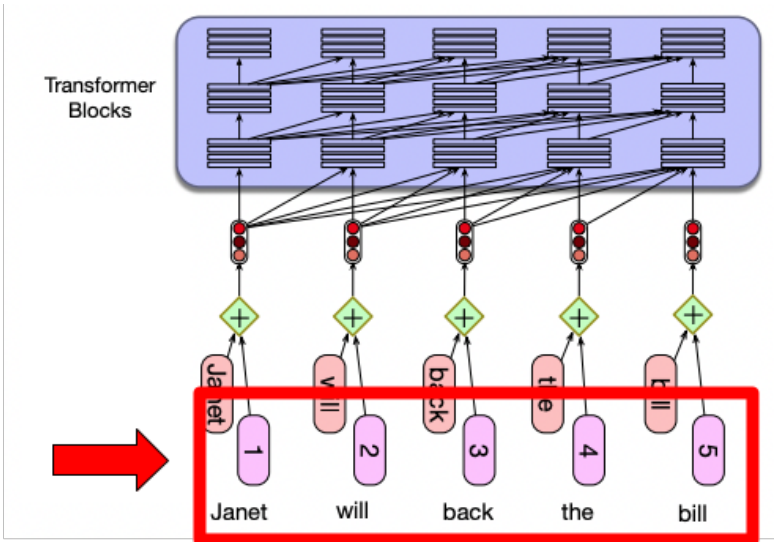


The above image visualizes the implementation of a part of Large Language Model (LLM). Write which part of LLM is displayed and explain the single most important reason why it helped to improve former language models.

Answer:

- Self attention or attention: 2p, any other description: 0p
- Explanation: It helps to capture long term dependencies: 3p
- Explanation: It helps efficiency or parallelization: 1p

14 LLM architecture (3 points)



Name the structure which is highlighted in the figure and describe how and why is it used (maximum 50 words).

Answer: Positional embeddings (1p), they are vectors added to token embeddings, encoding each token's position in a sequence (1p) and are needed to distinguish token positions and capture sentence structure (1p).

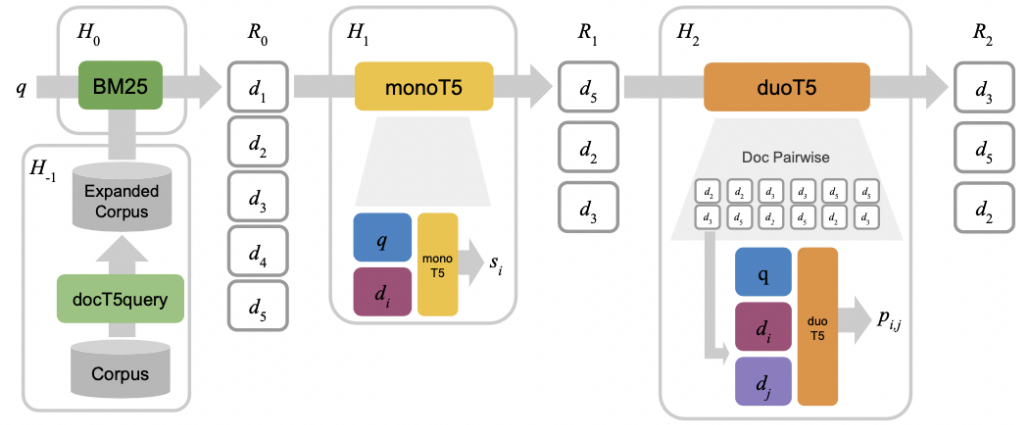
15 Neural IR (3 points; -1 if incorrect)

```python
# Load pre-trained Transformer-based model
model = LanguageModel.load_pretrained("transformer_model_name")

# Function to generate queries from a document
def generate_queries(document, num_queries=5):
    tokens = model.tokenize(document)
    embeddings = model.get_contextual_embeddings(tokens)

    # Use embeddings to generate query suggestions
    queries = []
    for _ in range(num_queries):
        query = model.generate_query(embeddings)
        queries.append(query)

    return queries


# Apply to multiple documents and index generated queries
def preprocessing_pipeline(corpus):
    query_index = {}
    for doc_id, document in corpus.items():
        queries = generate_queries(document)
        query_index[doc_id] = queries

    return query_index


# Execute preprocessing and save indexed queries
corpus = load_corpus("path_to_documents")
query_index = preprocessing_pipeline(corpus)
save_index(query_index, "query_index_path")
```

What is the name of the method implemented by the above pseudocode?

- ( ) DeepCT
- (X) Doc2Query
- ( ) Colbert
- ( ) SPLADE

16 Neural IR (6 points)



Briefly describe (provide name and typical implementation) of each stage in the above architecture.

Answer:

- Stage 1: initial ranking, BM25 (possibly with query or document expansion) or other sparse retrieval model: 2p for name and implementation
- Stage 2: pointwise reranking of top N documents (where N << size of the collection) implemented using BERT or T5 model: 2p for name and implementation
- Stage 3: pairwise reranking of top M documents (where M ¡ N) implemented using BERT or T5 model: 2p for name and implementation

17 Conversational Information Access (3 points)

Assuming a conversational search that employs a dialogue state architecture, which of the following are appropriate as agent-side intents?

- [ ] QUERY: Express information need
- [ ] CLARIFY: Expand on the information need by answering a clarification question
- [X] ANSWER: Provide an answer to the query
- [X] NO_ANSWER: Express that the request cannot be answered
- [ ] FEEDBACK: Provide feedback on the system's response
- [X] INQUIRE: Request additional information from the user

Scoring:

- 3p if all correct
- 1p for each correct answer, -1p for each incorrect answer; minimum 0p

18 Entity retrieval (2 points)

Suppose you wanted to compute PageRank on a knowledge graph to measure the importance of entities. Which of the following RDF predicates may be used to represent the entity graph?

- [ ] <foaf:name>
- [X] <dbo:wikiPageWikiLink>
- [X] <dbo:birthPlace>
- [ ] <rdfs:label>

Scoring:

- 2 points if all correct
- 1p for each correct answer, -1p for each incorrect answer; minimum 0p

19 Entity retrieval (12 points)

We want to create a fielded text-based document representation for the entity "Roger Federer" given the information associated with him in a knowledge base. We use four fields:

- names (name variants of the entity)
- types (type or category memberships)
- attributes (all literal objects that are not already in names)
- inlinks (all incoming relations, i.e., subjects where the given entity stands as object)

```
1 <dbr:Federer>                  <dbo:wikiPageRedirects>     <dbr:Roger_Federer>
2 <dbr:Roger_Federer>            <dbo:country>               <dbr:Switzerland>
3 <dbr:2007_US_Open_(tennis)>    <dbo:championInSingleMale>  <dbr:Roger_Federer>
4 <dbr:Roger_Federer>            <dct:description>           "Swiss tennis player"
```

For each RDF triple from the above image, select which field it should be mapped to (if any), and which preprocessing techniques should be applied (if any).

RDF triple	Mapped to	URI resolution	SPO⇒OPS flipping
1	names	Yes	Yes
2	NONE	Yes	No
3	inlinks	Yes	Yes
4	attributes	No	No

20 Entity retrieval (2x2 points)

Assuming a document d with the content "a b c a d c a a e" (where letters indicate terms), compute the following values to be used in the feature functions of the Sequential Dependence Model (SDM) for the query q="c a".

- Ordered bigram matches $c_o("c","a",d)$: [2]
- Unordered bigram matches in a window of $w = 3$ $c_u("c","a",d)$: [5]

Ordered matches:

1. a b **c a** d c a a e

2. a b c a d **c a** a e

Unordered matches in a window of 3:

1. **a** b **c** a d c a a e
2. a b **c a** d c a a e
3. a b c **a** d **c** a a e
4. a b c a d **c a** a e
5. a b c a d **c** a **a** e

21 Entity linking (3 points)

```python
# For each mention as key, the value is a dictionary with entityID as key
# and count as value.
SurfaceFormDict = dict[str, dict[str, int]]


def construct_surface_form_dict(
    mentions: list[tuple[str, str]]
) -> SurfaceFormDict:
    """Creates a surface form dictionary from a list of mentions.

    Args:
        mentions: List of (entityID, surface form) tuples.

    Returns:
        A surface form dictionary.
    """
    # TODO: Implement this function.


if __name__ == "__main__":
    # Usage example
    mentions = [
        ("Obama", "Barack Obama"),
        ("Obama", "Obama"),
        ("Obama", "president Obama"),
    ]
    print(construct_surface_form_dict(mentions))
```

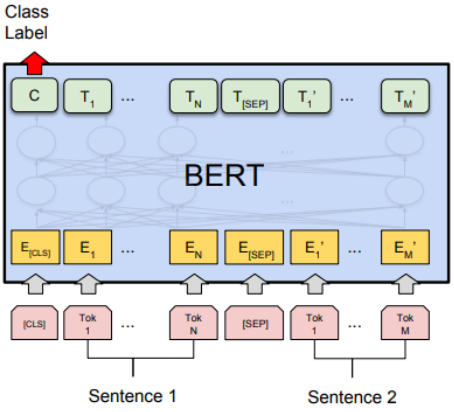
Write the body of the above Python method that constructs a surface form dictionary from an input list of entity mentions.

Scoring: 1p for each test case.

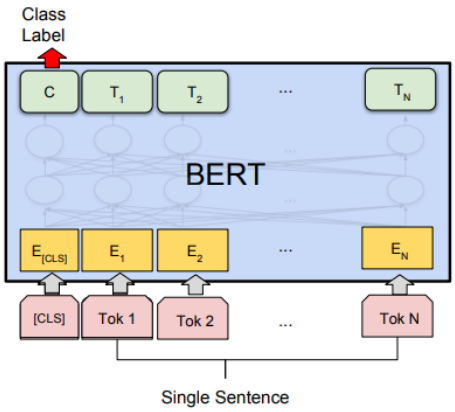
Sample solution and test cases:

```python
def construct_surface_form_dict(mentions: list[tuple[str, str]]) -> SurfaceFormDict:
    surface_form_dict = defaultdict(lambda: defaultdict(int))

    for entity_id, surface_form in mentions:
        surface_form_dict[surface_form][entity_id] += 1

    return surface_form_dict


# Test cases

# 1) Single entity-mention pair
mentions1 = [("entityID", "surface form")]
assert construct_surface_form_dict(mentions1) == {
    "surface form": {"entityID": 1},
}

# 2) Multiple entities for the same mention
mentions2 = [
    ("entityID1", "surface form"),
    ("entityID2", "surface form"),
    ("entityID3", "surface form"),
    ("entityID1", "surface form"),
]
assert construct_surface_form_dict(mentions2) == {
    "surface form": {"entityID1": 2, "entityID2": 1, "entityID3": 1},
}

# 3) Multiple entities and mentions
mentions3 = [
    ("entityID1", "entity"),
    ("entityID1", "Entity 1"),
    ("entityID2", "entity"),
    ("entityID3", "E3"),
    ("entityID3", "Entity 3"),
    ("entityID2", "entity"),
]
assert construct_surface_form_dict(mentions3) == {
    "entity": {"entityID1": 1, "entityID2": 2},
    "Entity 1": {"entityID1": 1},
    "E3": {"entityID3": 1},
    "Entity 3": {"entityID3": 1},
}
```

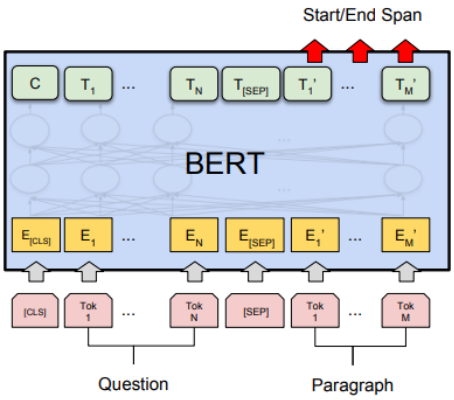
22 Classification (3 points; -1 if incorrect)



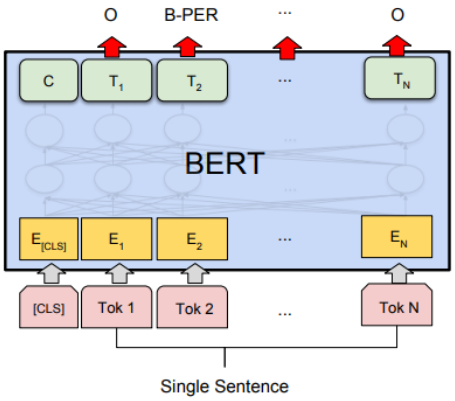
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

See the above image and select the BERT setup which you would use to classify a food recipe to 'vegetarian', 'vegan', and 'other' categories.

Answer: B

23 Classification (3 points; -1 if incorrect)

	Positive	Negative
intriguing	0.15	0.05
plot	0.07	0.06
but	0.08	0.09
confusing	0.08	0.14
ending	0.07	0.08

$$P(y|x) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Assume the above likelihoods for each word being part of a positive or negative book review, and equal prior probabilities for each class. **What class will Naive Bayes assign to the sentence 'Intriguing plot, but confusing ending.'?** Use unigram likelihoods and above Naive Bayes formula without smoothing.

- (X) Positive
- ( ) Negative

24 Recommender systems (3 points)

```python
def pearson_correlation(u: list[int], v: list[int]) -> float:
    """Calculate the Pearson correlation coefficient between two rating vectors.

    The ratings are assumed to be integers. Missing ratings are represented by 0.

    Args:
        u: A list of integers representing the ratings of user u.
        v: A list of integers representing the ratings of user v.

    Returns:
        A float representing the Pearson correlation coefficient between the two
        rating vectors.
    """
```

Write the body of the above Python function that computes the Pearson correlation coefficient between two rating vectors.

You may assume that the input is correct and that the two input vectors are non-empty and of equal size. Use the formula from the lecture slides.

Scoring: 1p for each test case.

Sample solution and test cases:

```python
def pearson_correlation(u: list[int], v: list[int]) -> float:
    numerator = 0
    denominator_u = 0
    denominator_v = 0
    nonzero_u = len([1 for r in u if r])
    nonzero_v = len([1 for r in v if r])
    # Need to have some items rated
    if not nonzero_u or not nonzero_v:
        return float("-inf")
    avg_u = sum(u) / nonzero_u
    avg_v = sum(v) / nonzero_v
    for r_u, r_v in zip(u, v):
        if r_u and r_v:
            numerator += (r_u - avg_u) * (r_v - avg_v)
            denominator_u += (r_u - avg_u) ** 2
            denominator_v += (r_v - avg_v) ** 2
    # Need to have some common items rated
    if denominator_u == 0 or denominator_v == 0:
        return float("-inf")
    return numerator / math.sqrt(denominator_u * denominator_v)


# Test cases

# 1) No common items rated (any extreme value or NaN is acceptable)
assert pearson_correlation([1, 0, 0, 0], [0, 0, 3, 2]) == float("-inf")
# 2) Perfectly correlated ratings
assert pearson_correlation([5, 0, 1, 0], [5, 0, 1, 0]) == 1.0
# 3) Negative correlation
assert pearson_correlation([1, 0, 4, 2, 0], [5, 0, 1, 2, 0]) == pytest.approx(-0.891, 0.1)
```

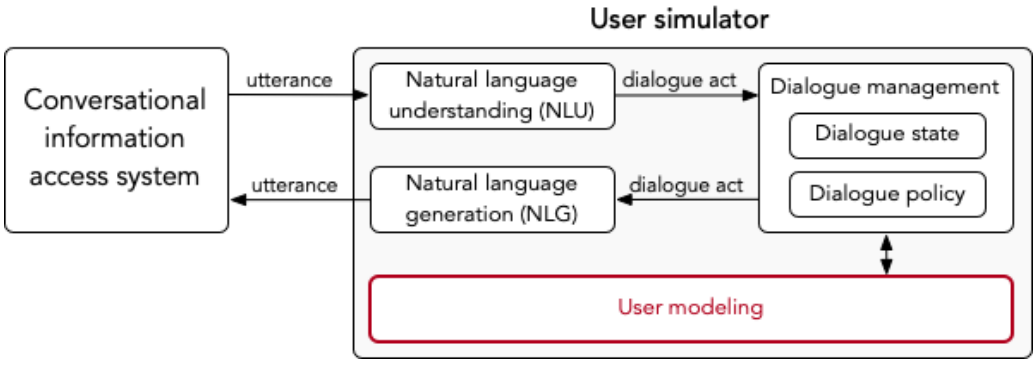
25 Recommender systems (2 points, -1 if incorrect)

Imagine that you are operating an online dating app, where registered users set up a profile with information about themselves, their interests, etc. Which of the following recommender approaches would be most appropriate in this scenario to recommend potential matches?

- ( ) Popularity-based
- (X) Content-based
- ( ) Collaborative filtering

Note: Collaborative filtering is not a good option because of the cold start problem (new users would not have any interaction data and could thus not receive any recommendations).

26 User simulation (6 points)



Suppose you want to create a user simulator agent that can interact with a conventional recommender system, using a modular architecture shown above. **Design a high-level API for the user modeling component, i.e., a UserModel class, by listing the class variables and signatures of the public methods that would be needed.** Specifically, you need to model

1. **User preferences** (in textual form, based on what the user has stated so far in the conversation)
2. **Historical ratings** on items (likes/dislikes)
3. **User patience** (a numerical value, which gets adjusted internally in this class based on the interactions with the system)

Use Python syntax. Note that you do not need to implement this functionality, just the provide method signatures and class variables. Use meaningful names for the variables and methods. Docstrings are only required if the method signature is not self-explanatory.

Scoring:

- Storing user preferences (as a string or list) (1p); methods for getting and setting user preferences (1p)
- Storing historical ratings (as a dict) (1p); methods for getting and setting historical ratings (1p)
- Storing user patience (as an int or float) and having a getter (1p); adjusting user patience based on interactions (e.g., last system dialogue act or entire conversation history) (1p); no points for adjusting based on a fixed delta amount provided as an argument

Sample solution:

```
class UserModel:
    def __init__(self, patience: float) -> None:
        self.preferences: list[str] = []
        self.ratings: dict[str, int] = {}
        self.patience = patience

    def get_preferences(self) -> list[str]:
        pass

    def add_preference(self, preference: str) -> None:
        pass

    def get_ratings(self) -> dict[str, int]:
        pass

    def get_item_rating(self, item_id: str) -> int:
        pass

    def add_rating(self, item_id: str, rating: int) -> None:
        pass

    def get_patience(self) -> float:
        pass

    def update_patience(self, history: DialogueHistory) -> None:
        pass
```

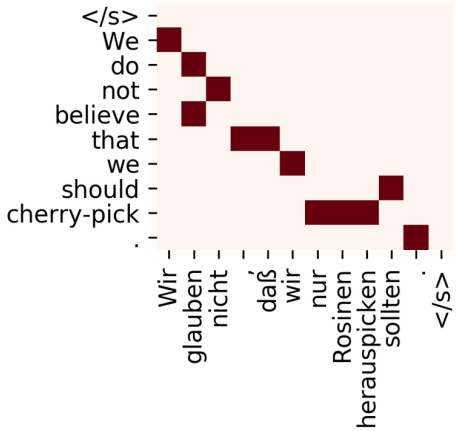
27 User simulation (4 points)

Consider the following statements about end-to-end user simulators. Indicate whether each statement is True or False.

	True	False
Typically require large datasets of historical user interactions to learn realistic behaviors	(X)	( )
Can directly learn complex user behaviors	(X)	( )
Preferred over modular simulators when interpretability is a requirement	( )	(X)
May struggle to generalize well to scenarios outside the training data	(X)	( )

Scoring: 1p each

28 CLIR (2 points)



Above is a translation alignment matrix between the English sentence 'We do not believe that we should cherry-pick' and its German translation 'Wir glauben nicht, daß wir nur Rosinen herauspicken sollten'. What is the translation of 'cherry-pick' according to this alignment?

Answer: "nur Rosinen herauspicken" (2p); any other solution: 0p

29 CLIR (3 points)

Explain a setup in cross-language retrieval in which query translation would be a better option than document translation (maximum 30 words).

- Correct answers: (3p)
  - We have large number of documents to translate, but only a small amount of queries.
  - We have queries in many different languages, but documents in a single (or small amount of) language.
  - Document collection is dynamic or constantly changing.
  - The translation quality of the query→document is better than the quality of document→query translation.
- Partially correct answers: (1p)
  - Queries are long enough and/or clear enough (not ambiguous)
  - Document set is large
- Anything else: 0p