**Recuperação de Informação**

Text Parsing and Tokenization

* 1. **1) Are the following statements true or false?**
  2. a) In a Boolean retrieval system, stemming never lowers precision. False.
  3. b) In a Boolean retrieval system, stemming never lowers recall. True.
  4. c) Stemming increases the size of the vocabulary. False.
  5. d) Stemming should be invoked at indexing time but not while processing a query. False.
  6. **2) Consider the tf-idf term weighting.**
  7. a) What is the *idf* of a term that occurs in every document? Compare this with the use of stop word lists.

It is 0. For a word that occurs in every document, putting it on the stop list has the same effect as idf weighting: the word is ignored.

* 1. b) Can the *tf-idf* weight of a term in a document exceed 1?

Sim

**3) Assume a biword index. Give an example of a document which will be returned for a query of *New York University* but is actually a false positive which should not be returned.**

Document=”Some alumni had arrived from New York. University faculty said that Stanford is the best place to study....”.  
É evidente que a New York University é uma frase específica, e refere-se a uma única coisa, no entanto ele devolve o resultado anterior, o que é definitivamente um falso positivo.

* 1. **4) Consider the Vector Space Model and classic Language Models:**
  2. a) How is the tokenization done in the VSM?

Tokenization is the process of breaking a text into individual tokens or terms. These tokens can be words, phrases, or other elements of the text. Tokenization can include

* Splitting the text into individual words or phrases using delimiters such as white space, punctuation, or other characters.
* Removing stop words
* Stemming or lemmatization, in order to reduce the dimensionality of the VSM and improve its effectiveness.
* Filtering or normalizing the remaining tokens to ensure that they are consistent and can be accurately compared. This can involve lowercasing all tokens, removing numbers or special characters, or other steps.
  1. b) How is a word token represented as a vector?

In a vector space model (VSM), each word token is represented as a vector in a high-dimensional space. The dimensions of the vector space correspond to the terms in the VSM, and the values in the vector indicate the presence or not of a given word in the document. For example, if a VSM contains the words "cat," "dog," and "mouse," a document that contains the word "cat" would be represented as a vector with a value of 1 for the "cat" dimension and 0 for the "dog" and "mouse" dimensions.

Classic language models, on the other hand, typically represent words as vectors in a lower-dimensional space, using a technique called word embedding. In word embedding, words are mapped to vectors in such a way that the vectors of similar words are close together in the vector space, while the vectors of dissimilar words are further apart.

* 1. c) Is it possible to compute the similarity between words in the VSM? Explain why.
  2. Yes. This is because the words in a VSM are represented as vectors in a high-dimensional space, and the similarity between two vectors can be calculated using a similarity measure such as cosine similarity. This measure calculates the cosine of the angle between two vectors, and the closer the vectors are to being parallel, the higher the cosine similarity will be.
  3. For example, if the VSM contains the words "cat" and "dog," and a document contains the word "kitty," the similarity between the vectors for "cat" and "kitty" could be used to determine that "kitty" is a synonym for "cat".
  4. d) How can you represent a word that has never been seen in the vector space model?
  5. By adding a new dimension to the vector space. This new dimension would correspond to the unknown word, and the vector for any document that contains the word would have a value of 1 for this dimension and 0 for all other dimensions.
  6. For example, if a VSM contains the words "cat," "dog," and "mouse," and a new document contains the word "rabbit," a new dimension for "rabbit" could be added to the vector space. The vector for this document would then have a value of 1 for the "rabbit" dimension and 0 for the "cat," "dog," and "mouse" dimensions.
  7. However, it should be noted that adding too many dimensions to the vector space can increase its complexity and reduce its effectiveness, so this approach should be used with caution.
  8. e) How is the word sequence guaranteed in the VSM/LM? Explain.

In a vector space model (VSM) or language model (LM), the word sequence is not necessarily guaranteed. A VSM or LM is a mathematical model that represents words or text in a continuous, numerical vector space, allowing for various mathematical operations to be performed on the words or text.

In a VSM or LM, words are typically represented as vectors of numbers, and the sequence of words in a document or text is not necessarily preserved. Instead, the relationship between words is captured by the similarity or distance between their corresponding vectors in the vector space. This means that the word sequence is not guaranteed in a VSM or LM, and the model is more focused on capturing the relationships between words.

* 1. **5) Transformer Language Models take a data-driven approach to text tokenization. Please explain:**
  2. a) How is BPE tokenization achieved?

Seja o vocabulário o conjunto de todos os caracteres individuais = {A, B, C, D,..., a, b, c, d....}

* 1. Normalmente é usado ainda o token “\_” in spaces.

1. Repetir:

- Escolher os dois símbolos que são mais frequentemente adjacentes no corpo de treino (dizer 'A', 'B')

- Acrescentar um novo símbolo fundido 'AB' ao vocabulário

- Substituir cada 'A' 'B' adjacente no corpus por 'AB'.

1. Fazer isto k iterações

BPE works by iteratively replacing the most frequent pair of bytes in the text with a single, unused byte. This process continues until a predefined vocabulary size is reached, at which point the resulting tokens can be used as input to the transformer model.

b) Stemming is a common element of whitespace tokenizers. Should it be also included in the BPE tokenizers?

It would be much faster, that’s a fact. But it would not make sense to use stemming in a BPE tokenizer because how BPE works is by pairing characters. By using stemming, words will be reduced to a “smaller” word with the same meaning, this way it will reduce the number of possible pairs the BPE tokenizer could make. Considering the purpose of BPE doesn’t make much sense.

* 1. c) How to represent a word that was never seen with BPE?   
     Depends if the combinations of characters of word was already seen. If not Is represent by the individual characters.
  2. d) How to represent a word that was never seen with the whitespace tokenizer?

One way to represent a word that was never seen with a whitespace tokenizer is to use a special token, such as <UNK>, to represent any unknown words. This token can be added to the list of words that the tokenizer uses to split the text, and it can be used to represent any words that the tokenizer encounters that it does not recognize.

Evaluation

* 1. **6) Consider an information need for which there are 4 relevant documents in the collection. Contrast two systems run on this collection. Their top 15 results are judged for relevance as follows (the leftmost item is the top ranked search result):**

1. **System 1 R N R N N N N N R N N N N N R**
2. **System 2 N R N N R N N N R N N R N N N** 
   1. a) What is the MAP of each system? Which has a higher MAP?

System 1 -> 0.25\* (1/1 + 2/3 + 3/9 + 4/15) = 0.57

System 2 -> 0.25\* (1/2 + 2/5 + 3/9 + 4/12) = 0.39  
System 1 has a higher average precision

* 1. b) Does this result intuitively make sense? What does it say about what is important in getting a good MAP score?

Sim porque no sistema 1, para a maioria dos casos, o k-documento relevante é retornado num rank melhor que o sistema 2. (Por exemplo, para cada k = 1 -> sistema 1 devolve logo na primeira posição enquanto que o sistema 2 só devolve na segunda o primeiro relevante). De notar que os primeiros documentos retornados são os que têm maior peso na fórmula.

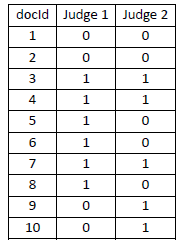
* 1. c) What is the Recall after 10 retrieved documents of each system?

System 1 -> ¾ = 0.75

System 2 -> ¾ = 0.75

* 1. d) Plot the precision-recall curve for both systems. Interpret the curve.
  2. São inversamente proporcionais.

**7) Below is a table showing how two human judges rated the relevance of a set of 12 documents to a particular information need (0 = nonrelevant, 1 = relevant). Let us assume that you’ve written an IR system that for this query returns the set of documents {4, 5, 6, 7, 8}.**



* 1. a) Calculate precision, recall, and F1 of your system if a document is considered relevant only if the two judges agree.

P = 2/5

Recall = 2/3

F1 = 2 / (1/P + 1/Recall) = 1/2

**8) Consider two ranking algorithms that for the same query produced the two following ranks:**

**S1: d4 d3 d5 d8 d2 and S2: d3 d9 d5 d6 d1**

* 1. a) Assuming that the relevant documents are d9, d1, d3 and d4, compute the precision and recall values of each system.

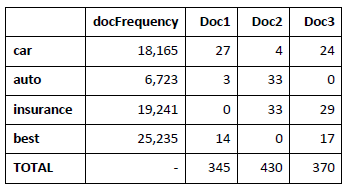
S1: P = 2/5 Recall = 2/4

S2: P = 3/5 Recall = 3/4

* 1. b) Assuming that the multi-value relevance judgments of documents are d9=1, d1=1, d3=3 and d4=2, assess and compare the two ranks with the appropriate metric. DCG
  2. c) Assume no relevance judgments and compare the two systems.

Podemos ver que o sistema 1 e 2 tem em comum o documento 3 (D3) podendo concluir que este documento é bastante relevante para a query.

Language Models and Retrieval models

* 1. **9) Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3. The collection contains 750,000 documents in total.**
  2. 
  3. a) Compute the tf-idf weights for the term’s car, auto, insurance, best, for each document.

Idf car: log(750000/18165) = 1.616

Idf auto: log(750000/6723) = 2.048

Idf insurance: log(750000/19241) = 1.591

Idf best: log(750000/25235) = 1.473

doc1 doc2 doc3 .

Tfidf car: 1.616 \* 27 = 43.6 1.616 \* 4 = 6.464 1.616 \* 24 = 38.784

Tfidf auto: 2.048 \* 3 = 6.14 2.048 \* 33 = 67.584 2.048 \* 0 = 0

Tfidf ins: 1.591 \* 0 = 0 1.591 \* 33 = 52.5 1.591 \* 29 = 46.139

Tfidf best: 1.473 \* 14 = 20.6 1.473 \* 0 1.473 \* 17

* 1. b) Compute the rank of the three documents for the query “auto insurance” on the vector space model.

Euclidian norm doc1: √(6.14^2 + 43.6^2 + 0 + 20.6^2) = V1

...

Doc1 = [43.6/V1, 6.14/V1, 0/V1, 38.784/V1] = []

...

Gerar a normal para dividir em baixo

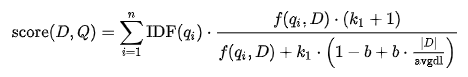
q=[0, 1\*2.048/2.593, 1\*1.591/2.593, 0] = [0, 0.7899, 0.614, 0]

Score(q, doc1) = Multiplicar vetor euclidian de q e doc1

Idém para DOC 2 e 3

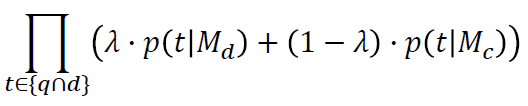
Ordenar os docs

* 1. **10) Consider the two following documents:**

1. **d1: Jackson was one of the most talented entertainers of all time**
2. **d2: Michael Jackson anointed himself King of Pop**
   1. a) Using a BM25 retrieval model determine which document is more relevant to the query q= “Michael Jackson” (consider b = 0.75 and k = 1.5).
   2. 

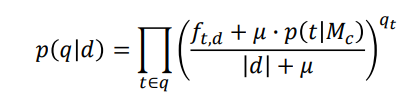
F(qi,D) -> quantas vezes o termo aparece no doc

Avgdl -> tamanho médio dos docs na coleção ((11 + 7) /2)

* 1. b) Using a language model with Jelinek-Mercer smoothing determine which document is more relevant to the query q= “Michael Jackson” (consider λ = ½)
  2. 

P(t|Md) -> nº de vezes q o termo aparece no doc / tamanho do doc

P(t|Mc) -> nº de vezes q o termo da query aparece na coleção/ tamanho da coleção

* 1. c) Using a language model with Dirichlet smoothing determine which document is more relevant to the query q= “Michael Jackson” (consider μ = 100)
  2. 

qt -> número de vezes que o termo aparece na query

f\_t,d -> numéro de vezes que o termo aparece no doc

**11) Show that models resulting from Dirichlet smoothing can be treated as probability distributions.  
That is, show that**

Let's consider a model that has been smoothed using a Dirichlet distribution with a concentration parameter α. This model will assign probabilities to each event in the model, represented by the probabilities p1, p2, p3, ..., pn.

The Dirichlet distribution is a distribution over n-dimensional probability vectors, and it can be shown that the sum of the elements of any such probability vector will always be equal to 1. In other words, the sum of the probabilities assigned to the events in the model will always be equal to 1.

c.q.d

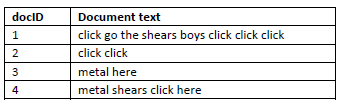
* 1. **12) Consider the Language Model with Jelineck-Mercer smoothing:**

a) What is the role of the lambda factor?

The lambda factor is used in Jelineck-Mercer smoothing to control the amount of weight given to the observed frequencies of words in the document versus the in the collection (corpus). In other words, it determines how much "smoothing" is applied to the language model. A higher lambda value means that more weight is given to the observed frequencies of words, while a lower lambda value means that more weight is given to the frequencies of words in the collection.

* 1. b) Why is smoothing necessary in Language Models?

Smoothing is necessary in language models because it helps to reduce the impact of rare or unseen words on the model's predictions. In a language model, we want to estimate the probability of a word occurring in a given context, but if a word has never been seen before, its probability would be zero, which would have a negative impact on the overall probability of the sentence. Smoothing helps to alleviate this issue by assigning a non-zero probability to words that have not been seen before, which helps to improve the overall accuracy of the language model and improve the model's performance on rare or infrequent words.

1. **13) Suppose we have a collection that consists of the 4 documents given in the below table.** 
   1. 

**Build a query likelihood language model for this document collection. Assume a mixture model (LMJM) between the documents and the collection, with both weighted at 0.5. Maximum likelihood estimation (mle) is used to estimate both as unigram models.**

* 1. a) Work out the model probabilities of the queries “click”, “shears”, and hence “click shears” for each document, and use those probabilities to rank the documents returned by each query.
  2. Language models
  3. Uma imagem com texto

     Descrição gerada automaticamenteUma imagem com mesa

     Descrição gerada automaticamente
  4. 

Model probabilities of the queries (formula do LMJM)

Uma imagem com mesa

Descrição gerada automaticamente

* 1. b) What is the final ranking of the documents for the query “click shears”?
  2. doc4 > doc1 > doc2 > doc3

**14) You have discovered that documents in a certain collection have a “half-life” of 30 days. After any 30-day period a document’s prior probability of relevance p(r|D) is half of what it was at the start of the period. Incorporate this information into LMJM. Simplify the equation into a rank-equivalent form, making any assumptions you believe reasonable.**

The probability of relevance for a document is given by: p(r|D) = p(D|r) \* p(r) / p(D)

… where p(D|r) is the probability of the document given that it is relevant, p(r) is the prior probability of relevance, and p(D) is the prior probability of the document.

This function could take the form of an exponential decay, where the probability of relevance decays exponentially over time. For example, if we assume that the half-life of documents is 30 days, and that the initial probability of relevance for a document is p0, then the probability of relevance after t days can be expressed as:

p(r|D, t) = p(D|r) \* p0 \* (1/2)^(t/30) / p(D)

… where (1/2)^(t/30) is the decay factor that reduces the probability of relevance by half for every 30-day period.

This equation can be simplified into a rank-equivalent form by assuming that the probability of the document given that it is relevant and the prior probability of the document are constant for all documents in the collection.

Under these assumptions, the equation can be simplified as follows: p(r|D, t) = p0 \* (1/2)^(t/30)

This information can be used to rank the documents in a collection, with more recent documents being given higher rankings than older documents.

* 1. **15) Write one sentence each describing the treatment that the LM with Jelinek-Mercer smoothing gives to each of the following quantities. Include whether it is present in the model or not and whether the effect is raw or scaled.**
  2. a) Term frequency in a document
  3. The LM with Jelinek-Mercer smoothing includes term frequency in a document and scales it by a smoothing parameter.
  4. b) Collection frequency of a term

1. It does not include collection frequency of a term.
   1. c) Document frequency of a term
2. It includes document frequency of a term and scales it by a smoothing parameter.
   1. d) Length normalization of a term

It includes length normalization of a term and scales it by the length of the document.

Learning to Rank

* 1. **16) The learning-to-rank approach aims to learn a ranking function that best ranks documents for each query.**
  2. a) What is the input to a learning-to-rank algorithm?

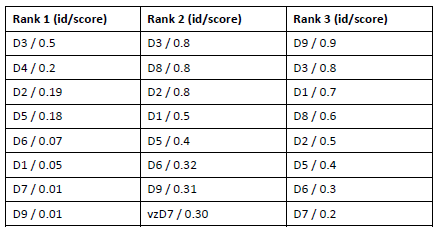
The input to a learning-to-rank algorithm consists of a set of queries, a set of documents for each query, and a set of relevance labels for the documents. The learning-to-rank algorithm uses this input to learn a ranking function that can be used to rank the documents for each query.

* 1. b) What is the role of the coefficients of the learning-to-rank model? What do they say about the role of each input feature?

The coefficients of the ranking function are used to represent the relative importance of the input features in determining the relevance of a document to a query. The coefficients indicate how much each input feature contributes to the overall relevance score for a document.

* 1. c) Learning to rank training data per label is highly skewed. In which ways can you compensate for the data unbalanced?
  2. One way is to use oversampling (increase the number of relevant documents) in the training set, or undersampling (decrease the number of irrelevant documents). Other way is using weighting methods, which assign higher weights to the minority class (i.e., the relevant documents) during training.
  3. d) The pointwise approach to learning-to-rank aims to rank documents by their importance to the input query in which way?
  4. Pointwise approach to learning-to-rank aims to rank documents by their importance to the input query by directly predicting the relevance of each document to the query and using the predicted relevance scores to rank the documents.

**17) Rank fusion methods combine ranks in different manners. Compute the fused ranks for the following three lists with the CombSUM, CombMNZ, BordaFuse and RRF.**



**CombSUM**

D1: Rank1 + Rank2 + Rank3 = 0.05 + 0.5 + 0.7 = 1.25

D2: Rank1 + Rank2 + Rank3 = 0.19 + 0.8 + 0.5 = 1.49

….

**CombMNZ**

D1: 3\*CombSUM = 3\*1.25 = 3.75

D2: 3\*CombSUM = 3\*1.49 = 4.47

…..

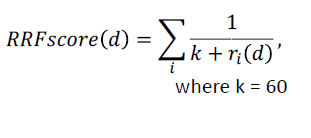
**BordaFuse**

Fórmula Geral: Di: Para cada rank (Número de documentos – posição do Di) e depois somar os valores todos.

D1: (8-6) + (8-4) + (8-3) = 11

D2: (8-3) + (8-3) + (8-5) = 13

**RRF**

**D1:**

**Nota:** Quando o rank não tem o documento ignora-se simplesmente.

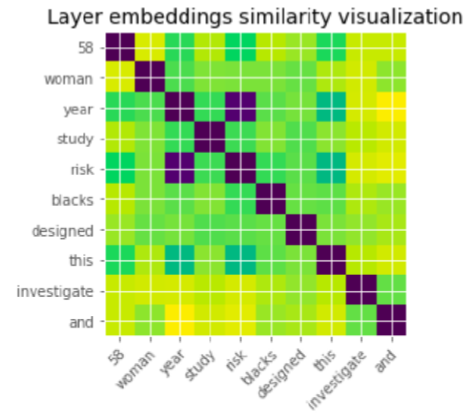
Contextual Embeddings and Self-Attention

* 1. **18) Consider the self-attention mechanism introduced by the Transformer.**
  2. a) Explain what is self-attention?
  3. Self-Attention compares all input sequence members with each other, and modifies the corresponding output sequence positions. This mechanism allows each word in the sentence to look at other words to better know which word contribute for the current word.
  4. b) How is the attention between two words computed?

The attention between two words is computed by comparing the corresponding word vectors using a dot product, and the resulting value is used to compute a weighted sum of the word vectors, which gives the output for each word.

* 1. c) The attention value between two tokens is used in which way?
  2. The attention value between two tokens is used as the weight for the corresponding value vector in the weighted sum. This weight indicates how much the output for each token should be influenced by the other token's vector. A high attention value indicates that the two tokens are related and that the output for each token should be strongly influenced by the other token's vector, while a low attention value indicates the opposite.
  3. d) How are the output embeddings of the self-attention layer computed?

The output embeddings of the self-attention layer are computed using a weighted sum of the input vectors. The weights for the sum are determined by the attention values between the input vectors, which are computed using a dot product.

* 1. **19) Consider the contextual embeddings computed by the Transformer encoder.**
  2. 
  3. a) In the Transformer architecture, what is it that the embeddings of layer 0 represent?
  4. In the Transformer architecture, the input sequence is first converted into individual tokens and passed through an embedding layer to convert them into dense vectors. The embeddings of layer 0 represent the dense vectors that correspond to the individual tokens in the input sequence. These dense vectors capture the semantic meaning of the tokens and are used as input to the rest of the Transformer model.
  5. b) In the Transformer architecture, what is it that the embeddings of layer 12 represent?

In the Transformer architecture, the output of the last layer is typically a sequence of dense vectors that represent the meaning of the input sequence in a high-dimensional space. The exact nature of these vectors depends on the specific implementation of the Transformer model and the task it is being used for. For example, in a Transformer model used for language translation, the vectors in the last layer may represent the meaning of the input sequence in the target language. In a Transformer model used for sentiment analysis, the vectors in the last layer may represent the overall sentiment of the input sequence.

c) The similarity matrix depicted next illustrates the similarity between layer 0 and layer 12. Why is the diagonal close to zero?

* 1. In a layer embedding similarity visualization matrix, the diagonal usually consists of the dot product of the embedding vectors with themselves. Because the dot product of a vector with itself is equal to the square of the vector's magnitude, the diagonal elements will always be close to zero if the embedding vectors are standardized (i.e., have a magnitude of 1).
  2. d) How is the text tokenization done in the Transformer architecture? Is it possible to represent a word that has never been seen before?

In the Transformer architecture, the input text is typically tokenized using a word tokenizer or subword tokenizer. A word tokenizer splits the input text into individual words, while a subword tokenizer splits the input text into subword units, which can include partial words and individual characters. Because a subword tokenizer can represent words as sequences of individual characters, it can handle out-of-vocabulary words by breaking them down into their constituent subwords. For example, if the word "transformer" has never been seen before, a subword tokenizer might represent it as the sequence of characters "trans", "form", and "er".

* 1. e) What is the role of the CLS token?
  2. In the Transformer architecture, the CLS token is used to represent the beginning of a text sequence. The Transformer model uses the representation of the CLS token as a summary representation of the entire input sequence.

**20) Consider the contextual embeddings as computed by the Transformer.**

* 1. a) Does the BERT Transformer maintain sequence information? If yes, how?
  2. Yes. In the BERT model, the input text is first split into individual tokens, and then these tokens are passed through an embedding layer to convert them into dense vectors. These vectors are then passed through multiple transformer layers, each of which processes the input in a different way.
  3. The BERT Transformer maintains sequence information by using a technique called "self-attention". In self-attention, each token in the input sequence is represented as a query vector, and the entire input sequence is represented as a set of key and value vectors. The query vectors are compared with the key vectors using a dot product, and the resulting values are used to compute a weighted sum of the value vectors, which gives the output for each token. This process allows the BERT model to attend to different parts of the input sequence at different times, which allows it to maintain and use the sequence information.
  4. b) What is the neighborhood of a word embedding vector in layer 0?
  5. It will be the words embeddings of tokens with closest meaning. In the Transformer architecture, the neighborhood of a word embedding vector in layer 0 is the set of other word embedding vectors in the same layer that are used to compute the output for that word.
  6. c) How can you compute the similarity between words in the Transformer?
  7. We can compute it with cosine similarity, which is defined as the cosine of the angle between the two vectors. This method gives a value between 0 and 1, where 0 indicates that the vectors completely unrelated and 1 indicates that the vectors are identical. (cos(0) = 1 and cos(90) = 0)

d) How to interpret the token embeddings visualization?

To interpret a token embedding visualization, you can look at the distances and orientations between the points to see how the vectors are related to each other. For example, if two points are close together and oriented in the same direction, it suggests that the corresponding vectors are similar and may be related in some way. On the other hand, if two points are far apart and oriented in opposite directions, it suggests that the corresponding vectors are dissimilar and may be unrelated.

Question Answering

* 1. **21) Consider the common QA processing pipeline.**
  2. a) In the QA architecture that was discussed in the course, what type of data is required? How should the data be pre-processed?

QA models are trained on large collections of input text questions. The data can be processed by using techniques such as tokenization, stopword removal, lemmatization, stemming.

* 1. b) Identify the two stages of a QA pipeline and explain their function.

Question processing is the stage responsible to detect the question type, answer type, focus, relations, and rewrite it as a query to send to a search engine.

Answer processing is the stage responsible to extract candidate answers and compute the answer.

* 1. c) How does the Transformer solve each part of the QA pipeline?

For example, in the data pre-processing stage, the Transformer can be used to encode the input text into a numerical representation that can be fed into the QA model. In the model prediction stage, the Transformer can be used to identify relevant passages in the input text and extract answer spans, or to generate answers based on the content of the input text. In the postprocessing stage, the Transformer can be used to refine and post-process the predictions generated by the QA model.

Live Systems Development

* 1. **22) Suppose that your boss asks you to develop a test collection to replace an existing corporate search engine. The company wants the test collection to be useful for the next 2-3 years.**
  2. a) Describe how you would build or acquire the different test collection components, and how much data is required for each component.

To build a test collection for a corporate search engine, I would follow the steps outlined below:

1. Identify the types of documents that will be included in the collection. This might include internal documents, such as product specifications, as well as external documents, such as scientific papers relevant to the company's industry.
2. Acquire a large corpus of these types of documents. This could be done by collecting all relevant documents.
3. Develop a set of queries that are relevant to the company and its employees. This could be done by gathering a sample of real-world queries that have been submitted to the company's existing search engine, as well as by identifying common types of queries that are likely to be used by employees in the future.
4. Create relevance judgments for the queries and documents in the collection.

It is generally best to have a large and diverse collection of data in order to provide a robust evaluation of model performance.

* 1. b) Detail the process of selecting queries and acquiring the corresponding relevance judgments. Your answer needs to be practical, i.e., no magic, and your budget isn’t infinite.

First, I would conduct research to understand the specific types of information that users of the corporate search engine are likely to be seeking. This might involve analyzing search logs to identify common queries, surveying users to understand their information needs, or conducting other forms of research to gather data on user behavior and preferences.

Based on this research, I would then create a set of queries that are representative of the overall set of topics that users are likely to be interested in. This would involve brainstorming and discussion to come up with a range of possible queries, and then refining and testing the queries to ensure that they are clear, specific, and relevant to the intended topics.

Once the queries have been selected, I would then need to acquire relevance judgments for each query. This would involve recruiting a group of evaluators who are familiar with the company's information sources and who can provide judgments on the relevance of the queries to those sources. The evaluators would use a standardized rating scale or other method to provide their judgments, and their responses would be used to train and evaluate the performance of the search engine.

Overall, this process would involve conducting research to understand user information needs, carefully selecting and crafting queries, and acquiring relevance judgments from evaluators. By following this process, I would be able to create a test collection that is well-suited to the needs of the company and that will be useful for the next 2-3 years.