**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.

Idf será 0, pois idf = log |D|/documento que contêm o termo e log(1) = 0, logo |D|/ documento que contêm o termo = 1, c.q.d. Como as stop words ocorrem na maior parte dos documentos então também terão um idf perto ou igual a zero.

* 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.**

New York e York University.

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?
  3. b) How is a word token represented as a vector?
  4. c) Is it possible to compute the similarity between words in the VSM? Explain why.
  5. d) How can you represent a word that has never been seen in the vector space model?
  6. e) How is the word sequence guaranteed in the VSM/LM? Explain.
  7. **5) Transformer Language Models take a data-driven approach to text tokenization. Please explain:**
  8. 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 “\_” para terminar as palavras.

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

b) Stemming is a common element of whitespace tokenizers. Should it be also included in the BPE tokenizers?  
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. Therefore, the BPE tokenizer is not going to work how it should.

* 1. c) How to represent a word that was never seen with BPE?
  2. d) How to represent a word that was never seen with the whitespace tokenizer?

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

* 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 ). De notar que os primeiros documentos retornados são os que têm maior peso na fórmula.

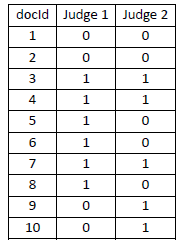
* 1. c) What is the Recall after 10 retrieved documents of each system?

System 1 -> ¾ = 0.75

System 2 -> ¾ = 0.75

* 1. d) Plot the precision-recall curve for both systems. Interpret the curve.

**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 = 2/8 Recall = 2/4

S2: P = 3/8 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.

S1: DCG = 2^3-1 / log(1+2) + 2^2-1 / log(1+1) = 7,45

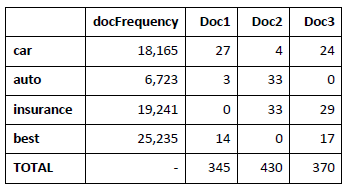
S2: DCG = 2^1-1 / log(1+2) + 2^1-1 / log(1+5) + 2^3-1 / log(1+1) = 8,02

Nota: log de base 2

* 1. c) Assume no relevance judgments and compare the two systems.

Podemos ver que o sistema 1 e 2 tem em comum o documentos 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 1.616 \* 24

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 + 0^2) = 6.14

Euclidian norm doc2: √(67.584 ^2 + 52.5 ^2) = 85.579

Euclidian norm doc3: √(0 ^2 + 46.139 ^2) = 46.139

Nota: So esta feito para os termos auto e insurance

Doc1 = [6.14/6.14; 0/6.14] = [1; 0]

Doc2 = [67.584/85.579; 52.5/85.579] = [0.7897; 0.613]

Doc3 = [0/46.139;46.139/46.139] = [0; 1]

Norma -> raiz quadrada((1 \*2.048)^2 + (1\*1.591)^2) = 2.593

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

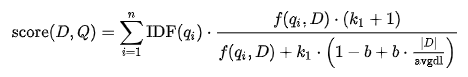
Score(q, doc1) = 0.7899\*1 + 0.614\*0 = **0.7899**

Score(q, doc2) = 0.7899\*0.7897 + 0.614\*0.613 = **1**

Score(q, doc3) = 0.7899\*0 + 0.614\*1 = **0.614**

**RANK:** Doc2 -> Doc1 -> Doc3

* 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(q1,D) -> quantas vezes o termo aparece no doc

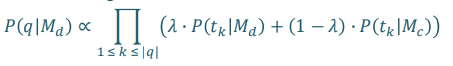
Avgdl -> tamanho médio dos docs na coleção ((11 + 7) /2 )

idf (q1 = "Michael") = log(N/n(q1)) = log(2/1) = 0.30

idf (q2 = "Jackson") = log(2/2) = 0

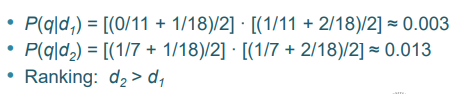
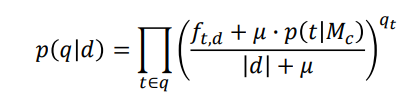
score(d1, q) = idf(q1) \* (0 \* 2.5)/(0+1.5\*(1-0.75+(0.75\*11/9) + idf(q2) \* (1 \* 2.5)/(1+1.5\*(1-0.75+(0.75\*11/9) = 0

score(d2, q) = idf(q1) \* (1 \* 2.5)/(1+1.5\*(1-0.75+(0.75\*7/9) + idf(q2) \* (1 \* 2.5)/(1+1.5\*(1-0.75+(0.75\*7/9) = 0.333

* 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(tk|Md) -> nº de vezes q o termo da query aparece no doc / tamanho do doc

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

* 1. 
  2. c) Using a language model with Dirichlet smoothing determine which document is more relevant to the query q= “Michael Jackson” (consider μ = 100)
  3. 

qt -> ASSUMI que é o tamanho da query

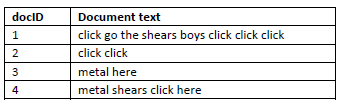
f\_t,d -> ASSUMI que é a frequencia do termo no doc

p(q|d1) = ( [(0 + 100 \* 1/18)/(11 + 100)]^2 ) \* ( [(1 + 100 \* 2/18)/(11 + 100)]^2 ) = 0.0025 \* 0.012 = 2.98e-5

**11) Show that models resulting from Dirichlet smoothing can be treated as probability distributions. That is, show that Σ𝑀𝑑𝑢(𝑡)𝑡=1.**

* 1. **12) Consider the Language Model with Jelineck-Mercer smoothing:**
  2. a) What is the role of the lambda factor?   
     Lambda is used for the process of parameter tuning for achieving a balance between underfitting and overfitting.
  3. b) Why is smoothing necessary in Language Models?

The term smoothing refers to the adjustment of the maximum likelihood estimator of a language model so that it will be more accurate.

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 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.**

* 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. c) Term frequency in a document
  3. d) Collection frequency of a term
  4. e) Document frequency of a term
  5. f) Length normalization of a term

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?

Para uma consulta q temos n documentos D = {d₁, …, dₙ} a serem classificados por relevância. Os elementos xᵢ = (q, dᵢ) são as entradas do nosso modelo.

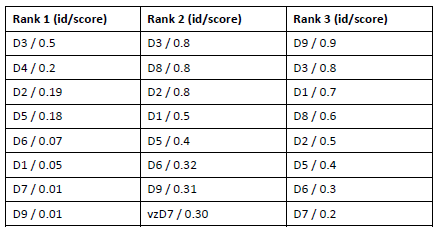
* 1. b) What is the role of the coefficients of the learning-to-rank model? What do they say about the role of each input feature?
  2. c) Learning to rank training data per label is highly skewed. In which ways can you compensate for the data unbalanced?

Nós usamos K-fold Cross-Validation para compensar a informação unbalanced

* 1. d) The pointwise approach to learning-to-rank aims to rank documents by their importance to the input query in which way?

Pointwise approache examina um documento de cada vez na loss function. Ele usa um único documento e treinam um classificador / regressor nele para prever o quão relevante é para a consulta atual. A classificação final é obtida simplesmente classificando a lista de resultados por essas pontuações de documentos.

**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: 0.05 + 0.5 + 0.7

D2: 0.19 + 0.8 + 0.5

**CombMNZ**

D1: 1/3\*(0.05 + 0.5 + 0.7)

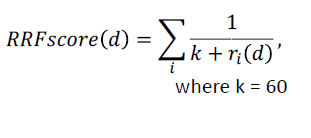
D2: 1/3\*(0.19 + 0.8 + 0.5)

**BordaFuse**

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

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

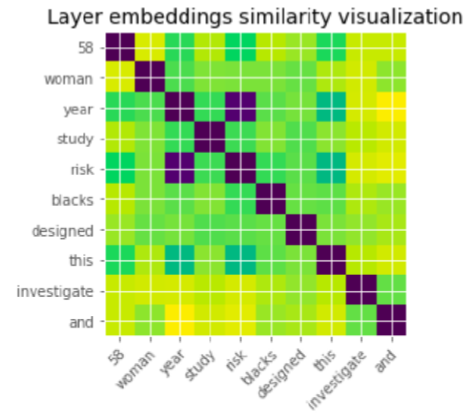
**RRF**

**D1:**

Contextual Embeddings and Self-Attention

* 1. **18) Consider the self-attention mechanism introduced by the Transformer.**
  2. a) Explain what is self-attention?
  3. Is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence
  4. b) How is the attention between two words computed?

The attention weights are calculated by normalizing the output score of a feed-forward neural network described by the function that captures the alignment between input at j and output at I.

* 1. c) The attention value between two tokens is used in which way?
  2. Is used to represent the similarity between that token pair
  3. d) How are the output embeddings of the self-attention layer computed?
  4. **19) Consider the contextual embeddings computed by the Transformer encoder.**
  5. 
  6. a) In the Transformer architecture, what is it that the embeddings of layer 0 represent?
  7. Representam os valores dos tokens (vetor com tantas posições quanto dimensões) após as primeiras operações.
  8. b) In the Transformer architecture, what is it that the embeddings of layer 12 represent?
  9. Representam os valores dos tokens (vetor com tantas posições quanto dimensões) após todas as operações.
  10. O parse da “sentence” fica completo e os valores de relevância dos tokens são finais.
  11. c) The similarity matrix depicted next illustrates the similarity between layer 0 and layer 12. Why is the diagonal close to zero?
  12. Os valores da diagonal estarão perto de 0 pois na layer 12, já se fez bastante processamento de tokens, enquanto que os tokens da layer 0 ainda não têm qualquer processamento. Desta forma os valores serão completamente diferentes.
  13. Se fosse um valor próximo de 0 signficava que nenhum processamento tinha sido feito.
  14. d) How is the text tokenization done in the Transformer architecture? Is it possible to represent a word that has never been seen before?
  15. e) What is the role of the CLS token?
  16. [CLS] is a token that will be used to predict whether or not Part B is a sentence that directly follows Part A.

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

* 1. a) Does the BERT Transformer maintain sequence information? If yes, how?

Sim, pois ele vai usa os valores de output de uma layer como input para a layer seguinte.

* 1. b) What is the neighborhood of a word embedding vector in layer 0?

It will be the words embeddings of tokens with closest meaning.

* 1. c) How can you compute the similarity between words in the Transformer?
  2. We can compute by the cosine-similarity.

d) How to interpret the token embeddings visualization?

Palavras marcadas perto umas das outras têm o mesmo significado no contexto da frase.

A posição no referencial não é relevante, o que é relevante, é a distância entre palavras.

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?
  3. b) Identify the two stages of a QA pipeline and explain their function.
  4. 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.
  5. Answer processing is the stage responsible to extract candidate answers and compute the answer.
  6. c) How does the Transformer solve each part of the QA pipeline?

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.
  3. 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.

Apontamentos

* 1. Stemming é o processo de reduzir palavras flexionadas ao seu tronco, base ou raiz.

Tokenization is the process of breaking up a given text into units called tokens. Tokens can be individual words, phrases or even whole sentences. In the process of tokenization, some characters like punctuation marks may be discarded. The tokens usually become the input for the processes like parsing and text mining.