Busca utilizando vetores densos:
Um exame do uso de queries artificiais e codificadores de transformers na tarefa de busca de documentos

Rafael Gonçalves Thomás Portugal

{ra186062,ra187646}@fee.unicamp.br

24 de junho de 2020

Abstract – This article explores the possibility of using neural models for the tasks of document retrieval and passage ranking. The authors explores specifically the architecture Two Tower, based on two BERT based transformers and a method for reducing documents to lists of queries called doc2query. These methods combined aims to create dense representational vectors for both documents and queries and then use a distance metric to calculate its similarity. This is an approach of great potential due to its reduction in the amount of computation required on the go - inference of dense vector of query and later calculation of its similarity with pre-computed document vectors.

Keywords – Deep learning, Natural language processing, Document retrieval, Passage ranking

1 Introdução

Um problema muito pesquisado na área de processamento de linguagem natural (PLN) é o de busca de documentos. Nele, há uma pergunta (ou *query*) e uma coleção de documentos que podem responder ou não a essa pergunta. O algoritmo deve retornar os documentos mais relevantes (e possivelmente ordenados em ordem de relevância). A maior dificuldade deste problema se dá pelo fato de que, em geral, a coleção de documentos é muito grande, e a operação deve ser feita em tempo limitado.

Os algoritmos utilizados comercialmente hoje, ainda usam características que dependem da equivalência de palavras (*exact match*, *bag of words*). O exemplo mais emblemático é o algoritmo BM-25 (ROBERTSON; ZARAGOZA, 2009).

O advento do aprendizado profundo e, mais especificamente, de modelos neurais aplicados a problemas de PLN, apresentou modelos neurais como alternativas interessantes por proverem uma representação plurívoca de elementos linguísticos. O uso de *word vectors* (MIKOLOV et al., 2013) – e de outras representações por vetores densos na forma de *embeddings* – no problema de busca de documentos se mostra uma estratégia interessante por não se limitar a utilizar a correspondência exata de palavras entre documentos e perguntas. Essas técnicas permitem, portanto, que se capture relações semânticas que não são levadas em conta em mecanismos de buscas tradicionais.

2 Trabalhos anteriores

Nossa abordagem se baseou em outros trabalhos recentes na área de processamento de linguagem natural, sobretudo em avanços na tarefa de busca de documentos (*document retrieval*) e ranquamento de passagens (*passage ranking*).

Uma das estratégias que mais contribuíram para nosso trabalho, foi a *doc2query* (NOGUEIRA et al., 2019), que consiste em usar um modelo neural para criar uma lista de perguntas, com base em um

documento de referência, visando expandir o documento com palavras que poderiam aparecer numa possível busca e, assim, aumentando as chances de ser encontrado.

Outro trabalho imprescindível para nosso projeto foi a arquitetura Two Tower (CHANG et al., 2020), baseada em dois transformers acoplados a uma mesma função de custo utilizada na tarefa de busca de documentos.

Por fim, nosso treinamento se inspirou na estratégia proposta em (KARPUKHIN et al., 2020), utilizando uma função de custo baseada no produto interno dos *embeddings* gerados pelo modelo.

3 Metodologia

Nesse trabalho usamos duas estratégias baseadas em modelos neurais. A primeira é o modelo *doc2query* que gera *queries* artificiais que podem ser respondidas por um determinado documento. Assim, os documentos foram substituídos por essas *queries* na entrada do modelo. A vantagem é que elas possuem tamanho de sequência consideravelmente menor que os documentos e, possivelmente, seriam mais semelhantes às *queries* presentes no conjunto de dados. Essas queries artificiais estão disponíveis em (NOGUEIRA et al., 2019) e não foi necessário gerá-las mais uma vez para nossos experimentos.

Além disso, usamos um modelo com arquitetura Two Tower (CHANG et al., 2020) que consiste em dois codificadores baseados em BERT (DEVLIN et al., 2018): um responsável por gerar vetores referentes às *queries* e outro aos documentos, ambos com a mesma função de custo compartilhada.

3.1 Treinamento

Na etapa de treinamento utilizamos os dados disponíveis já organizados em trios: cada amostra era composta pelo ID de uma *query* (qid), o ID de uma passagem relevante para atender à *query* (pid) e o ID de uma passagem irrelevante (nid). Com esses IDs, são obtidos os textos referidos através de uma tabela de *look-up*. Esses textos foram inseridos nos respectivos modelos para se obter suas representações em vetores densos. Nossa função de custo depende da similaridade de cosseno entre os vetores correspondente às *queries* e às passagens relevantes (positivas) e irrelevantes (negativas) como descrito em 1. Com $\text{sim}(a, b)$ dada pela equação 2 e ϵ sendo um número pequeno para evitar problemas de computação numérica.

$$\text{loss} = -\log\left(\frac{e^{\text{sim}(qid,pid)}}{e^{\text{sim}(qid,pid)} + e^{\text{sim}(qid,nid)}}\right) \quad (1)$$

$$\text{sim}(a, b) = \frac{a \cdot b}{\max(\|a\|_2 \cdot \|b\|_2, \epsilon)} \quad (2)$$

3.2 Validação

Para a validação. utilizamos um conjunto de dados em que, para cada *query*, há 1000 documentos relacionados obtidos através do mecanismo supracitado BM-25. Esses documentos são ranqueados utilizando a função de similaridade 2. Para medir a efetividade do modelo, utilizamos a *mean recyprocal rank* aplicadas sobre 10 posições (MRR@10) que avalia em um conjunto de 10 documentos ordenados, qual foi a posição dos documentos mais relevantes.

4 Conjunto de dados

Utilizamos o dataset MSMarco (MICROSOFT, s.d.) para fazer os experimentos. Este é um *dataset* multipropósito criado pela Microsoft utilizando resultados do mecanismo de busca BING. O dataset pode ser utilizado em tarefas de perguntas e respostas, extração de frases chaves e compreensão de textos por máquina no geral. Na tarefa escolhida, o conjunto de dados conta com cerca de 8 milhões de documentos e 1 milhão de *queries*. Cada *query* pode ter um 1, 2 ou nenhum documento relevante associado.

Para a etapa de treinamento, os dados foram

utilizados na forma de trios contendo uma *query*, um documento relevante e outro irrelevante. Já na validação, foi utilizado um conjunto que associava cada *query* a uma lista com 1000 documentos, entre eles a passagem relevante.

5 Experimentos

Nos experimentos realizados, foram utilizadas 1 milhão de amostras de treinamento e cerca de 100 no conjunto de validação. O tamanho do batch utilizado foi de 50 para o treinamento e 2 para validação. Essa discrepância se deve à quantidade de vetores criados por amostra em cada caso: para treinamento eram 3 vetores por amostra, enquanto na validação eram 1001 vetores por amostra.

Os vetores densos construídos foram de dimensão 128. As sentenças foram truncadas em 32 tokens, que é o tamanho médio de uma *query* para o nosso conjunto de dados (ou seja, cada documento foi representado por apenas 1 *query* artificial).

O otimizador utilizado foi o Adam, com alguns valores de learning rate. Para testar as variações, foram feitos experimentos que incluíam: tentar forçar o *overfit* em um *batch* e fazer o treinamento com uma época e validação. Os gráficos produzidos se encontram nas figuras abaixo.

Observamos que, para os valores testados, a curva de aprendizado do *overfit* de um batch se estabilizou em valores menores ao diminuir a taxa de aprendizado (*learning rate*, lr), porém ainda distantes de zero (valor esperado para *overfit*). Nas situações de treino, a curva se mostrou bastante mais ruidosa e tende a se estabilizar em valores altos, mesmo com a presença de vales e a depender da taxa de aprendizado. O MRR obtido na validação em todos os experimentos foi zero, ou seja, em nenhuma dessas situações o modelo foi capaz de colocar os documentos relevantes nos 10 primeiros colocados para alguma *query*.

Dos 3 experimentos apresentados, obtivemos a melhor curva de aprendizado com $lr = 1e - 4$. Valores de lr menores, tenderam a convergir (ao menos no *overfit* de um *batch*), mas com menor velocidade. Os testes com lr maiores divergiram ou, como mostrado na figura 6, convergiram para valores absurdos (ainda piores do que os modelos 1 e 2).

5.1 Experimento 1: lr $= 1e - 6$

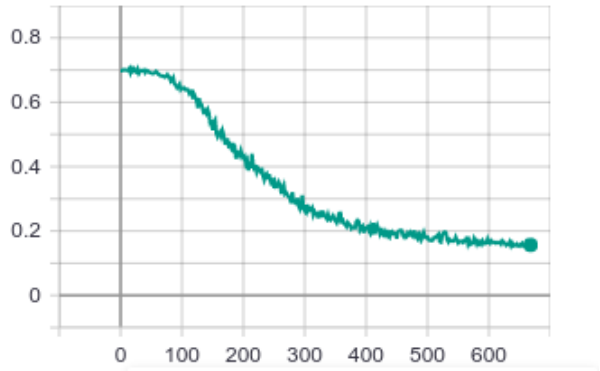

Figura 1: Curva de aprendizado da tentativa de realizar overfit sobre 1 batch do modelo 1.

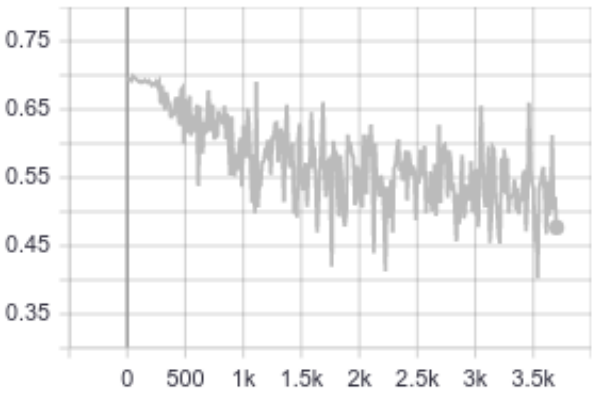

Figura 2: Curva de aprendizado do modelo 1.

5.2 Experimento 2: lr $= 1e-4$

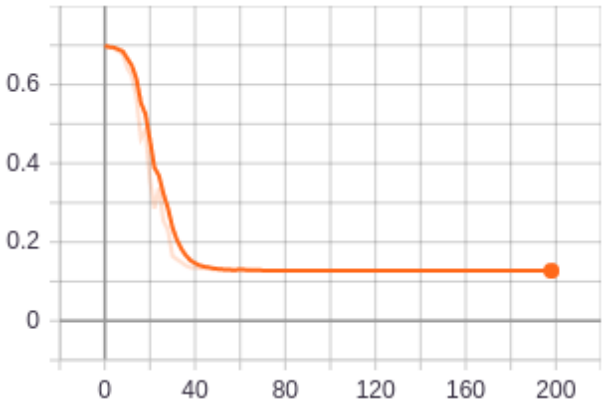

Figura 3: Curva de aprendizado da tentativa de realizar overfit sobre 1 batch do modelo 2.

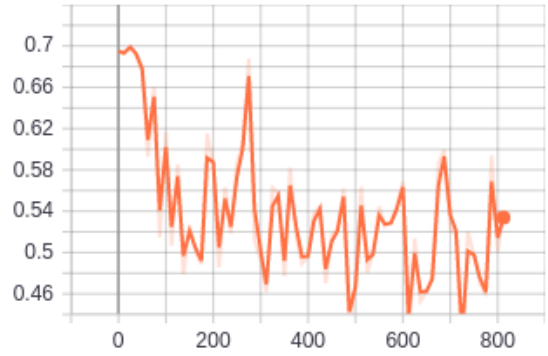

Figura 4: Curva de aprendizado do modelo 2.

5.3 Experimento 3: lr $= 1e-2$

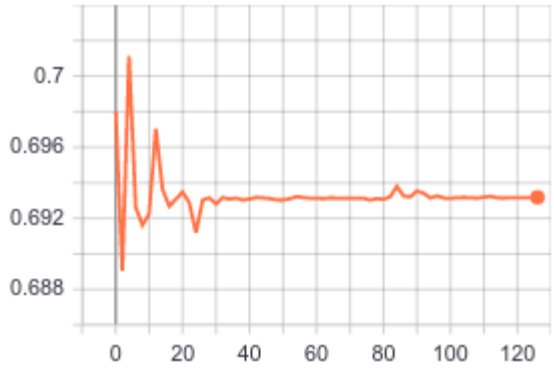

Figura 5: Curva de aprendizado da tentativa de realizar overfit sobre 1 batch do modelo 3.

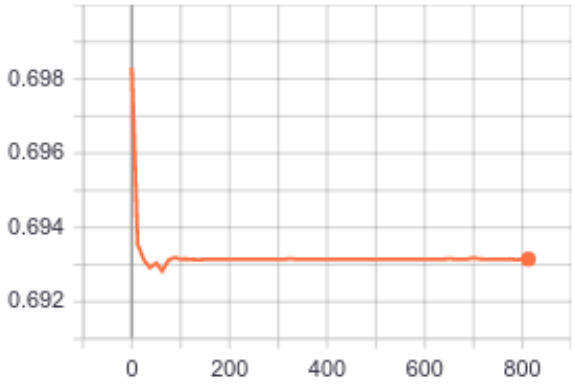

Figura 6: Curva de aprendizado do modelo 3.

6 Conclusões

Os resultados mostrados no presente artigo certamente não são definitivos, mas apontam para uma dificuldade em criar um modelo neural que seja capaz de ordenar os documentos baseados apenas em *queries* artificiais.

Nossa primeira hipótese é que a função de custo utilizada não foi capaz de representar bem o problema para qual o modelo foi proposto. Isso pode se dar, por exmemplo, pela métrica utilizada utilizar apenas um exemplo positivo e um exemplo negativo.

Outra hipótese possível é que ao limitar o tamanho da sequência em 32 tokens, ou seja, apenas 1 *query* artificial para cada documento, não houve informação o suficiente capturada pelos vetores densos.

7 Trabalhos futuros

Para trabalhos futuros, uma sugestão imediata seria utilizar mais sentenças negativas para computar a função de custo no treinamento. Essa prática foi realizada em (KARPUKHIN et al., 2020) e pode ser um dos motivos para o comportamento atípico da curva de aprendizado. Outra possibilidade seria experimentar com outras funções de custo diferentes, um exemplo dessa abordagem está presente em (NOGUEIRA; CHO, 2019).

No presente estudo, não foram feitos outros experimentos no que diz respeito à dimensão do vetor denso, algotitmo de otimização e tamanho máximo das sequência. Trabalhos futuros podem explorar outras combinações desses parâmetros que podem resultar no sucesso em resolver a tarefa já especificada.

Referências

CHANG, W.-C. et al. *Pre-training Tasks for Embedding-based Large-scale Retrieval*. 2020. arXiv: 2002.03932 [cs.LG].

DEVLIN, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. arXiv: 1810.04805 [cs.CL].

GONÇALVES, R.; PORTUGAL, T. *search-with-dense-vectors*. https://github.com/RafaelGoncalves8/search-with-dense-vectors. (acessado em 16/06/2020).

KARPUKHIN, V. et al. *Dense Passage Retrieval for Open-Domain Question Answering*. 2020. arXiv: 2004.04906 [cs.CL].

MICROSOFT. *MsMarco*. https://microsoft.github.io/msmarco/. (acessado em 16/06/2020).

MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

NOGUEIRA, R.; CHO, K. *Passage Re-ranking with BERT*. 2019. arXiv: 1901.04085 [cs.IR].

NOGUEIRA, R. et al. *Document Expansion by Query Prediction*. 2019. arXiv: 1904.08375 [cs.IR].

ROBERTSON, S.; ZARAGOZA, H. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.