TRANSFORMERS HIERÁRQUICOS PARA BUSCA DE ESPECIALISTAS NA ACADEMIA

RENNAN CORDEIRO LIMA

TRANSFORMERS HIERÁRQUICOS PARA BUSCA DE ESPECIALISTAS NA ACADEMIA

Segundo Projeto Orientado em Computação, elaborado como requisito para conclusão do curso de Ciência da Computação da Universidade Federal de Minas Gerais

Orientador: Rodrygo Luis Teodoro Santos

Belo Horizonte

Março de 2021

Cordeiro Lima, Rennan

D1234p *Transformers* hierárquicos para busca de especialistas na academia / Rennan Cordeiro Lima. — Belo Horizonte, 2021
ix, 19 f. : il. ; 29cm

Projeto Orientado a Computação — Universidade Federal de Minas Gerais
Orientador: Rodrygo Luis Teodoro Santos

I. Título.

CDU 519.6*82.10

Para Alécia, Cleber e Silvia

Resumo

A tarefa de busca de especialistas é problema de ranking: dado um tópico de especialidade, deseja-se construir uma lista ordenada por relevância de pesquisadores de maneira que, aqueles no topo da lista sejam os especialistas. O desafio, no contexto desse trabalho, é que os dados presentes são de uma entidade diferente: as publicações e incluem seus atributos como título, data de publicação e palavras-chaves e etc, assim a única informação disponível dos pesquisadores é a sua relação de autoria com esses documentos. Para resolver esse problema, esse trabalho utiliza modelos recentes de aprendizado profundo, em específico os *transformers*, que têm se mostrado melhor que as estratégias tradicionais de ranqueamento. Foi aplicado o conceito de *transformers* hierárquicos, no qual um modelo recebe uma sequência de publicações relacionadas a um autor de entrada a fim de gerar uma lista de representações densa a serem consumidas por um segundo modelo que deve ranqueá-las.

Palavras-chave: *Ranking*, Recuperação de Informação, Aprendizado profundo, Busca de especialistas.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Contextualização

A quantidade de informação acadêmica disponível na sociedade tem crescido cada vez mais, de maneira que todos os dias, diversos artigos, teses, dissertações, relatórios técnicos e todo tipo de literatura acadêmica são publicados. Junto a esse grande volume de dados, acumulam-se também os metadados da produção científica, como autoria, citações e etc.

Esses dados podem ser usados de diversas maneiras, como, entender tendências da comunidade cientifica, o que pode contribuir para entender melhor a produção científica global. Podem ser usados para descobrir o estado-da-arte para algum problema, para encontrar estratégias que auxiliem o desenvolvimento de outros trabalho acadêmicos e soluções do mercado, em diversas áreas. Contudo, devido a quantidade extrema de informação tem se tornado cada vez mais inviável que humanos a explorem manualmente, o que torna necessário desenvolver estratégias automatizadas para isso.

Uma tarefa em específico, nesse contexto, é o de buscar especialistas na acadêmia de acordo com um tópico de especialidade. Isso é relevante para diversas áreas e a própria academia pode se beneficiar desse dado: descobrir quem se destaca em cada área permite que os pesquisadores se conheçam e se conectem, o que aumenta o fluxo de conhecimento e acelera o avanço da ciência.

Outro cenário em que uma ferramenta de encontrar especialistas é extremamente útil é no jornalismo. Muitas vezes, os meios de imprensa, para criar uma reportagem, notícia e conteúdo jornalístico no geral, recorrem a pesquisadores, que atuam como fonte e, além agregar conhecimento ao fato relatado, também garante maior credibilidade perante o público. No cenário atual, repleto das conhecidas *fake news*, o problema da credibilidade tem ganhado atenção, logo, identificar as pessoas que possuem conhe-

cimento em uma área e traçar o perfil de algum pesquisador tem se tornado cada vez mais necessário.

1.2 Aprendizado profundo

Aprendizado profundo, ou também *Deep Learning*, se refere a uma família de modelos de aprendizado de máquina, na qual, são aprendidas representações do dado, sem que seja necessário extrair características. Esse tipo de técnica é o estado-da-arte para diversas áreas, como processamento de linguagem natural, processamento de áudio e vídeo. LeCun et al. [2015]

Estratégias clássicas de *ranking* são limitadas pela ocorrência exata de termos das consultas, como TF-IDF Mitra et al. [2017] Liu [2011]. Estratégias tradicionais de aprendizado também são limitadas LeCun et al. [2015], pois não utilizam os dados diretamente, apenas um que utilizam conjunto de características que os descrevem. Uma estratégia de aprendizado profundo, nesse trabalho consideraria o contexto de cada termo, o que ajudaria a discriminar especialistas dos demais pesquisadores.

Assim, a área de recuperação de informação também tem se beneficiado dos notáveis avanços em *deep learning* como pode ser visto em Yang et al. [2019], pois essa abordagem tem se mostrado melhor que as estratégias tradicionais de ranqueamento de documentos. Fica claro, portanto, que tarefa de encontrar de especialistas pode se beneficiar desse avanço.

1.3 Definição da tarefa

Nessa tarefa, há um tópico de especialidade, representado como um texto em linguagem natural, que servirá como consulta. Para essa consulta, deseja-se encontrar aqueles indivíduos com a maior relevância, que nesse contexto, pode-se interpretar como grau de especialidade.

Essa tarefa pode ser solucionada através da atribuição de um *score* numérico para todos os pesquisadores conhecidos em relação a uma consulta. Para isso, pode-se utilizar os dados associados a cada um desses indivíduos, que, no contexto desse trabalho, são os dados das publicações nas quais eles foram autores ou co-autores. Em seguida, pode-se ordenar esses pesquisadores em ordem decrescente por esse *score*, de maneira a criar um *ranking*. Com o ranking construído, pode-se assumir, então, que os pesquisadores nas primeiras posições, aqueles com os maiores *scores*, sejam os

especialistas. Assim, fica claro que a tarefa de busca de especialistas, em essência, é uma tarefa de ranking.

1.4 Projeto Orientado em Computação I

A motivação para esse trabalho é resultado do que foi explorado no primeiro projeto orientado a computação. Nele, a mesma tarefa foi explorada, porém, foi desenvolvida uma solução alternativa.

Essa solução pode ser descrita da seguinte maneira: para cada pesquisador, havia uma estratégia para gerar uma *string* que fosse capaz de incluir suas informações. Essa *string* era concatenada aos termos da consulta e esse texto era então usado como entrada para um modelo baseado em *transformers*. Esse modelo, tinha como saída a relevância daquele pesquisador para aquela consulta, assim, o modelo aprendia a pontuar os pesquisadores para as consultas.

A estratégia para gerar a *string* de cada autor se baseava na exploração das *keywords* de seus documentos associados. Nessa estratégia, para cada autor, eram extraídas as N *keywords* mais frequentes, que eram ordenadas pela frequência e em seguida concatenadas. A intuição por trás disso é que as *keywords* são da mesma natureza das consultas, ou seja, áreas do conhecimento humano e, portanto, poderiam ser a melhor maneira para representar a publicações e consequentemente os autores.

Essa solução foi validada através de experimentos e teve um desempenho superior ao *baseline* proposto no trabalho anterior. Dentre os experimentos feitos, verificou-se que, quanto mais *keyword* eram incluídas na *string* de um pesquisador, e consequentemente apresentadas ao modelo no treinamento, melhor era o desempenho da solução.

Contudo, embora tenha tido um bom resultado, essa solução possuía alguns problemas. O primeiro deles era o fato de que essa heurística de *keywords* não possuía um fundamento teórico sólido. Além disso, ela desconsiderava grande parte dos dados de cada publicação, como os títulos, resumos e as datas em que foram publicadas. Por fim, na estratégia adotada, era necessário alimentar o modelo baseado em *transformers* toda vez que fosse necessário avaliar a relevância de um pesquisador, seja em treino ou em teste (nesse caso, também seria num cenário de produção). Esse fato também era um problema, pois, esse tipo de arquitetura possuí um tempo de resposta alto (citar), o que tornaria inviável utilizá-los numa máquina de busca de pesquisadores aberta ao público.

1.5 Objetivo

A motivação para continuar explorando essa tarefa nesse trabalho POC2 advém da percepção desses pontos de melhoria da solução anterior. Por tanto, o objetivo anterior continua, ou seja, desenvolver uma abordagem para ranquear pesquisadores usando aprendizado profundo, para resolver a tarefa de encontrar especialistas na academia. Contudo, essa solução deve superar as limitações do trabalho anterior, em específico, objetiva-se utilizar mais e melhor os metadados da publicação e construir uma estratégia que possui um menor tempo de resposta. Essas medidas tornariam essa solução mais eficiente e eficaz.

Capítulo 2

Fundamentação teórica e trabalhos relacionados

2.1 Conceitualização

2.1.1 Learning to rank

Atualmente, as estratégias mais sofisticadas se baseiam em Learning to Rank (L2R). Essa categoria engloba diversas abordagens que utilizam aprendizado de máquina para tarefas de ranking como definido em Liu [2011]. Em específico, este trabalho tem como referencial o aprendizado supervisionado no escopo de Learning to Rank, no qual um conjunto de documentos e suas respectivas relevâncias para uma consulta em específico (também chamado conjunto de treino) é usado para otimizar um modelo neural, como descrito em Mitra & Craswell [2018] e Liu [2011]. Com o modelo treinado em mãos, é possível aplicá-lo num conjunto de testes Liu [2011], de modo a avaliar sua qualidade. A metodologia usada nesse trabalho seguem esses passos citados.

2.1.1.1 Abordagens de aprendizado

Há 3 maneiras de se ranquear documentos usando aprendizado supervisionado: Mitra & Craswell [2018] Liu [2011]:

- **Pointwise**: Os dados de entrada são organizado na forma documento-consulta e a saída é um valor numérico que representa a relevância geral do documento para a consulta.
- **Pairwise**: Os dados de entrada são organizado na forma documento-documento-consulta e a saída é um indicador (geralmente numérico) de qual documento de

entrada é o mais relevante.

- **Listwise**: Os dados de entrada são organizado na forma listadedocumentos-consulta e a saída é um valor que representa quão bem aquela lista está ordenada por relevância, geralmente usa-se métricas de avaliação de ranqueamento para isso.

Essas diferentes abordagens podem ser testada afim de encontrar aquela que tem melhor desempenho prático, o que é feito nesse trabalho.

2.2 Trabalhos relacionados

2.2.1 Busca de especialistas

Outros trabalhos já exploraram essa tarefa anteriormente. Resumidamente, Macdonald & Ounis [2011] utilizam de estratégias clássicas de ranking como TF-IDF e BM25 para classificar os documentos seguido de estratégias de votação para gerar *features* a serem usadas de entrada para um modelo neural. Cada conjunto de *feature* gerado representa uma entidade a ser classificada.

Diferentemente, Moreira et al. [2011] utilizaram diferentes características de cada pesquisador como entrada num modelo neural, as quais foram divididas em 3 grupos: o primeiro é formado pela similaridade textual entre a consulta e os documentos para cada um dos pesquisadores, semelhante a estratégia adotada em Macdonald & Ounis [2011]. Já o segundo grupo foi baseado em estatísticas de publicação de cada autor, como média de publicação por ano, número total de publicações etc, essas *features*, porém, são em sua maioria independentes da consulta. Por fim, o último grupo foi formado por características extraídas a partir do grafo autor-publicação, como H-index.

Vale lembrar que, partes das informações utilizadas em Moreira et al. [2011] não estão presentes no *dataset* usado nesse trabalho, como informações de conferencias e metadados dos autores, o que mostra uma limitação a ser superada pela solução apresentada pelo trabalho.

Pode-se observar que essas estratégias focam no aprendizado baseando-se na extração de características para cada um dos autores e através da agregação da relevância individual de cada documento para a consulta. Além disso, nenhum deles utilizam das estratégias mais recentes de processamento de linguagem natural. Portanto, há um potencial de melhorar o desempenho dessa tarefa através uma estratégia de ponta a ponta que utilize os atributos dos documentos diretamente como entrada, dessa forma

o trabalho de agregação é transferido para rede neural e por conta disso, também será aprendido.

2.2.2 Transformers

Dentre as estratégias mais recentes de processamento de linguagem natural, uma arquitetura de extrema relevância são os *transformers*, que se baseiam no que é conhecido como mecanismo de atenção Vaswani et al. [2017]. Nele, existem diversas *heads* que recebem todos os termos da entrada e são responsáveis por capturar diversas relações entre esses termos, de maneira que, os padrões aprendidos por esse modelo sempre consideram todos os termos. É por causa dessa capacidade que os *transfomers* têm se mostrados bastante poderosos.

Devido ao seu potencial, diversos trabalhos têm explorado a arquitetura de *transformers*. Devlin et al. [2018] mostraram em seu trabalho um modelo baseado em *transformers* chamado BERT. Além de demonstrarem que os *transformers* são capazes de resolver tarefas de processamento de linguagem natural, comprovaram que eles também são o estado da arte para diversas dessas tarefas, por exemplo, identificação de paráfrase. Ademais, Reimers & Gurevych [2019] mostraram que, quando bem treinados, os *transformers* são capazes de gerar representações vetoriais de sentenças que podem ser usadas em várias tarefas, incluindo aquelas da área de Recuperação de Informação.

Um trabalho em especial, no qual esse trabalho se baseia foi o desenvolvido por Pappagari et al. [2019]. O trabalho resolveu a tarefa de classificação de documentos da seguinte maneira: cada documento é dividido em suas diversas sentenças, em seguida, cada uma delas é apresentada a um modelo responsável por gerar seus vetores densos num mesmo espaço. Ao final, todos os vetores de um documento são apresentados a um segundo modelo que é responsável por rotular o documento usando as classes disponíveis no dado. Essa estratégia foi proposta para contornar a limitação dos *transformers* em relação ao tamanho da entrada, pois essa arquitetura, devido a sua complexidade quadrática, não permite a utilização de documentos com milhares de termos.

Essa noção de dividir o documento e utilizar diferentes chamadas de um *transformer* para representar cada uma das sentenças é chamado apresentado no artigo como *hierarquical transformers*, que nesse trabalho está sendo chamado de *transformers* hierárquicos. O paralelo com o problema de busca de especialista é feito através da noção de que um especialista é formado por uma sequência de publicações, assim como um documento é formado por uma sequência de sentenças. Logo, pode-se aproveitar esse conceito na tarefa de busca de especialista, e a maneira como a solução foi adaptada é apresentada com mais profundidade ao longo desse trabalho.

Capítulo 3

Solução proposta

Esse trabalho propõe a seguinte solução para o problema de busca de especialistas na academia: desenvolver uma arquitetura em duas etapas, a primeira é responsável por gerar representações densas de uma publicação, a partir de seus dados, e a segunda é responsável por agregar todas essas representações em torno de um autor de maneira a criar um único vetor denso para esse indivíduo. A consulta também deve ser representada no mesmo espaço, de maneira que, seu vetor seja próximos daqueles que indicam os autores especialistas para elas. Com ambas as etapas, pode-se atribuir um *score* para cada par consulta-autor possível, e, portanto, construir o ranking.

Assim, foi necessário desenvolver dois modelos, o primeiro, presente na primeira etapa é responsável por gerar tais vetores e o segundo por agregá-los. Nas subseções abaixo está o detalhamento melhor de cada uma dessas etapas

3.1 Representações densas de publicações

Inicialmente, seguindo o que foi feito por Reimers & Gurevych [2019], foi desenvolvido um modelo siamês capaz de criar vetores densos para cada uma das publicações. Esse modelo é baseado em *transformers* e pode ser descrito da seguinte maneira: um texto, que pode ser extraído de uma publicação ou de uma consulta, é dividido em seus termos. Cada um desses termos é uma entrada para o modelo de *embeddings* contextuais e a saída é o vetor denso de cada um deles. Em seguida, esses vetores são agregados através de um *average pooling*, de maneira a criar um só vetor que representem toda a sentença da entrada. Uma última camada densa, com menos dimensões do que o vetor denso, é adicionada ao final e é responsável por diminuir o tamanho das representações.

Em relação o treinamento desse modelo, como ele é siamês, é necessário apresentar pares de entrada que devem ter vetores próximos, de maneira que o modelo aprenderá

a representá-los com esse objetivo. No contexto desse trabalho, deseja-se aproximar representações de consultas das publicações às quais elas estão associadas. Assim, foi desenvolvido um conjunto de dados que possuam esses pares.

Na imagem abaixo, pode ser visto um diagrama da arquitetura desse primeiro modelo, com seus módulos e etapas.

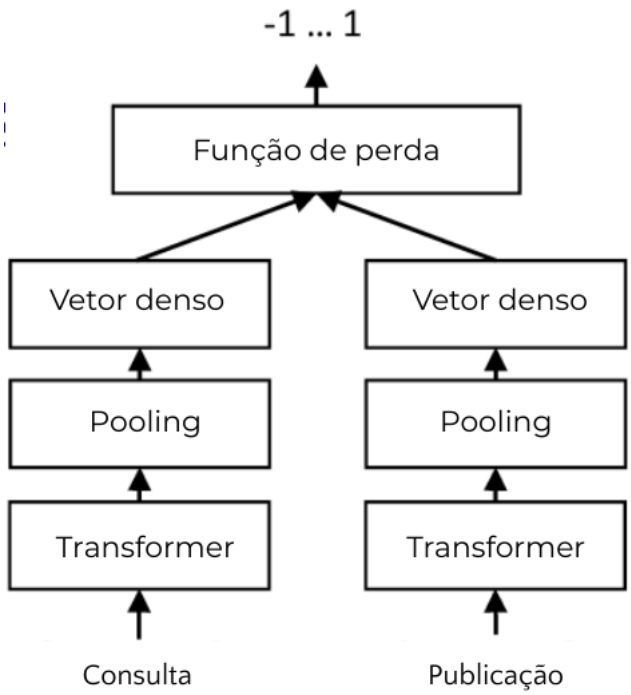

Figura 3.1: Diagrama da arquitetura usada

3.2 Agregação de representações em torno de um autor

A outra parte da solução é responsável por agregar vários vetores de publicações em torno de um autor a fim de criar um vetor, no mesmo espaço, que o represente. Para isso, é explorado o relacionamento autor-publicação está ilustrado pelo grafo bipartido abaixo em que cada aresta indica que um autor publicou um dos documentos. Podemos observar que há publicações conectadas a mais de um autor e autores conectados a mais de uma publicação, ou seja, os pesquisadores estão relacionado a um conjunto de documento.

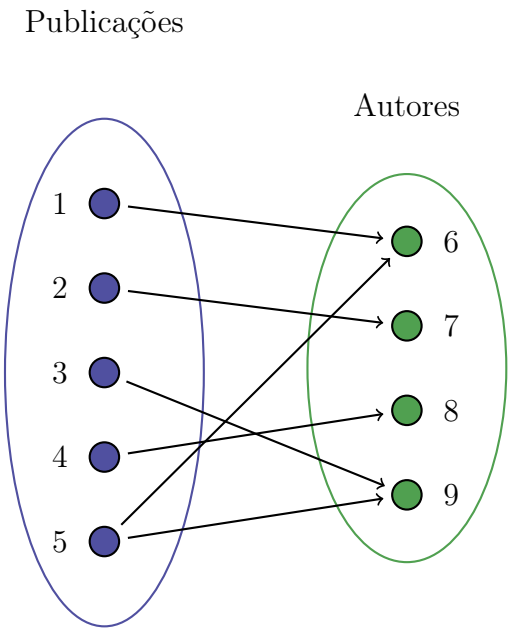

Figura 3.2: Grafo bipartido representando o relacionamento autor-publicação

Para isso, foi utilizada uma rede baseada em recorrência, a LSTM Hochreiter & Schmidhuber [1997]. Essa rede, recebe cada um dos vetores e retorna uma representação alterada deles. Foi escolhida para esse trabalho, pois, esse tipo de arquitetura lida bem com sequências, então, espera-se que também podem ser aplicada a uma sequência de publicações.

Além disso, na mesma arquitetura, foi adicionada uma camada densa, que recebe a saída da LSTM concatenada ao vetor que representa a consulta. Essa camada densa aprende a indicar o grau de relevância de determinado autor. Como o erro calculado da função de perda é propagado para todos os parâmetros, sejam da LSTM ou da camada densa, ao final, teremos uma rede capaz de indicar, para um conjunto de publicações e um autor, qual é a relevância para uma determinada consulta.

Vale lembrar que, nesse trabalho, o que define cada conjunto é o fato de que, todas as publicações possuem o mesmo autor em comum, porém, nada impede que esse conjunto seja definido por outros atributos entre as publicações, como ano em que foram publicadas, conferência onde foram publicadas e até mesmo a organização de seus autores. Ou seja, essa arquitetura poderia ser usada para aprender a relevância de vários tipos de entidade para um determinado tema, não se limita a pesquisadores.

Na imagem abaixo, pode ser visto um diagrama da arquitetura desse primeiro modelo, com seus módulos e etapas.

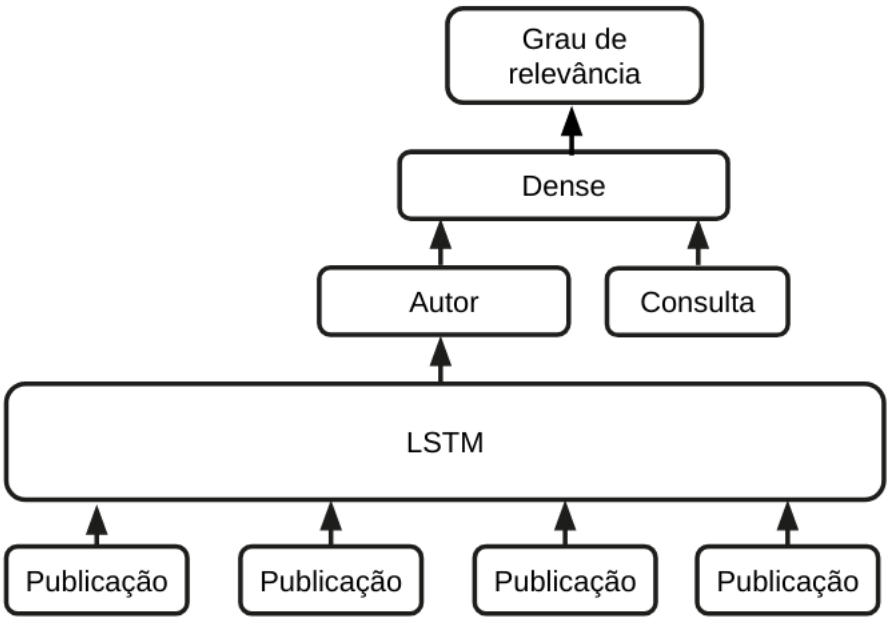

Figura 3.3: Diagrama da arquitetura usada

3.3 Autores candidatos

Outra etapa dessa estratégia é construir o conjunto dos candidatos a especialistas para cada uma das consultas. Embora idealmente deseja-se atribuir um *score* para todos os pesquisadores possíveis, o tempo de predição para cada um deles torna inviável construir o *ranking* de todos os pesquisadores presentes no dado para todas as consultas.

Por conta disso, é preciso contornar esse problema através da construção de um conjunto de pesquisadores candidatos para uma consulta, os quais serão os únicos apresentados ao modelo. Para isso, para cada uma das consultas, são recuperados os documentos mais relevantes, em seguida, para cada um dos pesquisadores presentes nesse conjunto, é atribuído um *score* calculado através da soma das relevâncias dos seus documentos presentes nos recuperados, processo semelhante ao que é feito no CombSUM Shaw & Fox [1994].

Por fim, todos os pesquisadores são ordenados do maior ao menor *score* e aqueles com os maiores N *scores* são apontados como os pesquisadores candidatos.

Capítulo 4

Configuração experimental

4.1 Baseline

O *baseline* escolhido para esse modelo se baseia na técnica CombSUM Shaw & Fox [1994] apresentada, na qual a relevância de um autor é a soma das relevâncias do conjunto intercessão entre seus documentos e os documentos recuperados pela consulta. Outras variações também foram testadas, como CombMNZ Belkin et al. [1995], porém tiveram um resultado inferior.

4.2 Dataset

O conjunto de dados escolhido chama-se LExR (Lattes Expertise Retrieval) Mangaravite et al. [2016] que inclui mais de 200 mil pesquisadores com doutorado de diversas áreas da ciência publicados na plataforma Lattes, além de todas as publicações dos pesquisadores escolhidos, totalizando mais de 11 milhões de publicações.

Os documentos são formados pelos metadados de cada uma das publicações presentes, por exemplo, a data da publicação, os autores de cada uma delas, e suas *keywords*. Porém, não são todas as publicações que incluem todos esses dados, pois apenas alguns campos estão presentes em todos os documentos, como o título, *abstracts*, por exemplo, estão presentes apenas em um subconjunto deles. Por conta disso, há uma limitação da quais atributos puderam ser usados.

Além dos metadados, o conjunto inclui a informação de especialidade da seguinte maneira: há um conjunto de pares pesquisador-tópico, e cada par está atrelado à informação da especialidade daquele pesquisador para aquele tópico. Há 3 graus de especialidade ordenados de 1 a 3. Os pesquisadores que não estão presentes no conjunto

desses pares não são especialistas para nenhum tópico, ou seja, possuem grau zero de especialidade para todos os tópicos.

4.3 Processamento dos dados

Inicialmente recuperou-se os 1000 pesquisadores mais relevantes como candidatos. Esse valor foi calculado através de experimentos, que revelaram cerca de 80% de revocação média, contra cerca de 40% utilizando os 100 maiores *scores*, além de incluir mais consultas com pelo menos um relevante recuperado. São esses 1000 pesquisadores - com suas respectivas publicações - que serão usados no modelo, tanto para treino quanto para teste.

4.4 Treinamento

Para o treinamento do modelo de representações, foram usados todos os pares *keyword*-título de publicação possíveis, que totalizaram 20 milhões de pares. Esses pares foram organizados de maneira a criar 20 milhões de entradas triplas, com um *keyword*, um título relevante e um título não relevante que foram alimentados ao modelo. Foi treinado apenas por uma época.

Em relação o segundo modelo, que recebe várias publicações, o treinamento foi *pair-wise*, para cada pesquisador eram escolhidos todos os outros pesquisadores da mesma consulta para formar todos os pares possíveis. Além disso, para cada relevante, foram selecionados mais dois pesquisadores irrelevantes para a consulta para formar mais pares. Foi treinado por 10 épocas.

4.5 Métricas

Com o modelo treinado, algumas métricas foram calculadas para avaliar a qualidade em relação ao *baseline*. A primeira delas é NDCG, *normalized discounted cumulative gain*, métrica clássica de *ranking* que considera níveis graduados de relevância, como temos nos dados. Foi calculado também *o* MRR, média dos *reciprocal rank* de cada consulta, o qual é definido pela razão entre um e a posição do primeiro elemento relevante. Por fim, *Success*, que indica o número de consultas com pelo menos um especialista recuperado.

4.6 Resultados

Resultados					
Solução	NDCG@5	Success@5	NDCG@10	Success@10	MRR
Baseline (BM25 + CombSum)	**0.1036**	9	**0.1674**	**21**	**0.2516**
Proposta pelo trabalho	0.0765	**11**	0.0913	11	0.1508

Tabela 4.1: Valores calculados das métricas para modelo e para o *baseline*

Pode ser observado pela tabela que o modelo, embora tenha tido um resultado melhor em *success* nas 5 primeiras posições do *rank* teve um desempenho pior nas 10 primeiras, e em todos os cortes nas outras métricas. Assim pode-se dizer que seu desempenho é inferior ao *baseline*.

No gráfico da Figura 4.1 abaixo, podemos ver a análise de métricas por consultas, cada uma das barras pode ser interpretada como a diferença do valor da métrica escolhida para uma consulta do modelo em relação ao *baseline*, portanto, idealmente, todas as barras devem ser positivas. Nesse caso, foi usado o NDCG paras as 10 primeiras posições do ranking e embora o gráfico mostre que em algumas consultas o modelo foi superior ao *baseline*, na maioria delas o *baseline* foi superior.

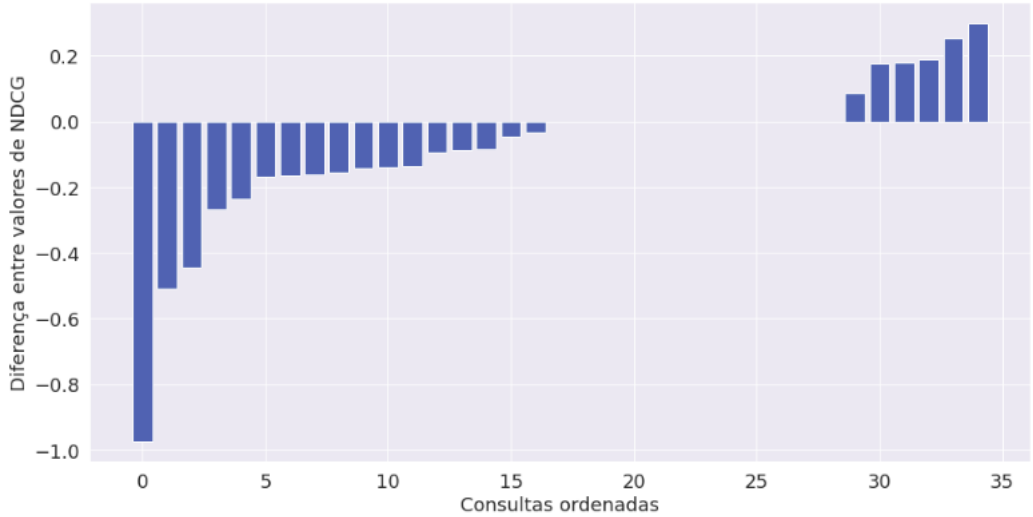

Figura 4.1: Diferença entre modelo e *baseline* por consulta - NDCG

O gráfico da Figura 4.2 abaixo é análogo ao anterior, porém a métrica usada nesse caso é de *recall*. Podemos observar um comportamento parecido, no qual o modelo é inferior ao *baseline* na maioria das consultas.

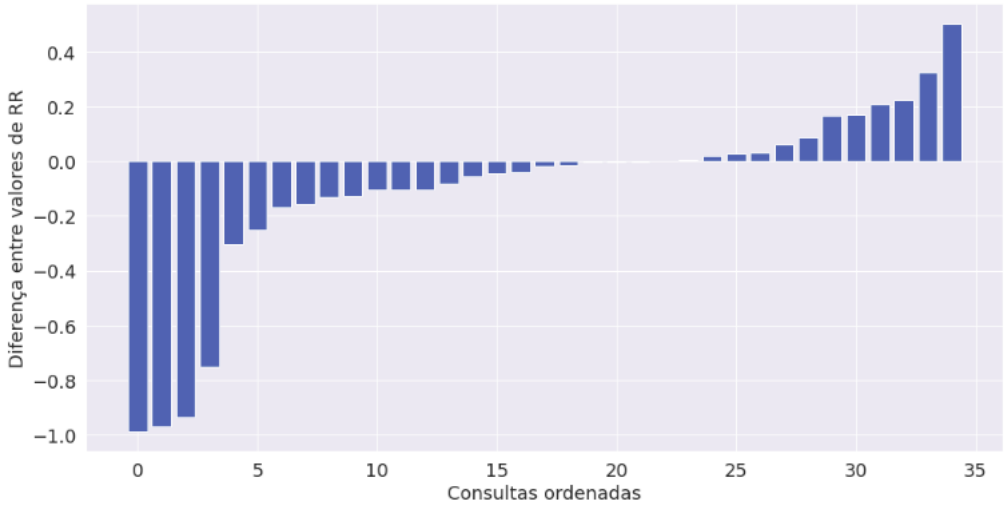

Figura 4.2: Diferença entre modelo e *baseline* por consulta - Reciprocal Rank

Foi feita uma análise qualitativa do desempenho individual de cada uma solução em relação as consultas. Enquanto a consulta "*biological activity*" e outras geraram um resultado pior no modelo proposto por esse trabalho, outras levaram a um melhor desempenho modelo em relação ao *baseline*. A tabela abaixo expõe alguns dessas consultas.

Consultas melhores no baseline	**Consultas melhores no que foi proposto**
rheumatic heart disease	herbaspirillum seropedicae
biological activity	sediment cores
electronic structure	inductively coupled plasma
herbaspirillum	text mining

Tabela 4.2: Exemplos de consultas melhores no modelo e no *baseline*

Capítulo 5

Conclusão

Nesse trabalho, foi desenvolvida uma solução que utiliza aprendizado profundo para treinar um modelos de *ranking* de pesquisadores. O trabalho explorou uma estratégia em duas etapas, representação de publicação e agregação de seus vetores em torno de autores. Nos resultados, no qual o modelo proposto pelo trabalho foi inferior ao *baseline* usado na maioria das métricas e seus respectivos cortes.

5.1 Trabalhos futuros

Como trabalhos futuros, diversas direções podem ser seguidas. Pode ser feita a análise mais aprofundada dos resultados das consultas, com mais métricas e com uma análise qualitativa de cada uma delas, com a verificação dos resultados. Além disso, pode-se utilizar mais consultas para teste, e entender melhor os resultados em mais cenários.

Ainda nesse ponto, para validar a robustez da solução, ela poderia ser testada em múltiplos *datasets* e usando todas as consultas para teste, através do *cross-validation*. Esse tipo de análise é essencial para entender o melhor o comportamento do modelo e consequentemente, os pontos fortes e fracos da solução, pode facilitar na identificação de como podem ser melhorados.

Seguindo esse tipo de análise, pode ser feitas, também, mudanças nas estratégias de treinamento e na arquitetura, como utilizar uma rede alternativa para agregar representações. Pode-se escolher também, outras estratégias de ordenação de publicações a serem apresentadas nos, mudanças no treinamento tanto da primeira quanto da segunda arquitetura. Não se descarta também, uma mudança maior na abordagem, com uma possível simplificação dos módulos.

Além disso, o trabalho se limitou a incluir apenas as publicações que o autor publicou em sua representação, porém, incluir outras publicações, como de seus autores

vizinhos, através da exploração do grafo autor-publicação, pode expandir as informações sobre um autor, o que poderia melhorar a solução. Outro tipo de informação que poderia ser agregada futuramente é a dos resumos, que não foi explorada no escopo desse trabalho.

Todas essas adaptações podem levar a um melhor desempenho da solução, principalmente em relação ao *baseline*, de maneira a torná-la mais competitiva. Além disso, elas poderiam ser validadas experimentalmente através do arcabouço já desenvolvido nesse trabalho.

Referências Bibliográficas

Belkin, N. J.; Kantor, P.; Fox, E. A. & Shaw, J. A. (1995). Combining the evidence of multiple query representations for information retrieval. Em *Proceedings of the Second Conference on Text Retrieval Conference*, TREC-2, p. 431–448, USA. Pergamon Press, Inc.

Devlin, J.; Chang, M.-W.; Lee, K. & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735--1780.

LeCun, Y.; Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Liu, T. (2011). *Learning to Rank for Information Retrieval*. Springer Berlin Heidelberg.

Macdonald, C. & Ounis, I. (2011). Learning models for ranking aggregates. Em Clough, P.; Foley, C.; Gurrin, C.; Jones, G. J. F.; Kraaij, W.; Lee, H. & Mudoch, V., editores, *Advances in Information Retrieval*, pp. 517--529, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mangaravite, V.; Santos, R.; Ribeiro, I.; Gonçalves, M. & Laender, A. (2016). The lexr collection for expertise retrieval in academia.

Mitra, B. & Craswell, N. (2018). An introduction to neural information retrieval. *Found. Trends Inf. Retr.*, 13:1–126.

Mitra, B.; Diaz, F. & Craswell, N. (2017). Learning to match using local and distributed representations of text for web search. Em *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, p. 1291–1299, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Moreira, C.; Calado, P. & Martins, B. (2011). Learning to rank for expert search in digital libraries of academic publications. Em Antunes, L. & Pinto, H. S., editores, *Progress in Artificial Intelligence*, pp. 431--445, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pappagari, R.; Zelasko, P.; Villalba, J.; Carmiel, Y. & Dehak, N. (2019). Hierarchical transformers for long document classification. *CoRR*, abs/1910.10781.

Reimers, N. & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. Em *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Shaw, J. A. & Fox, E. A. (1994). Combination of multiple searches. Em *THE SECOND TEXT RETRIEVAL CONFERENCE (TREC-2*, pp. 243--252.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u. & Polosukhin, I. (2017). Attention is all you need. Em Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S. & Garnett, R., editores, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yang, W.; Zhang, H. & Lin, J. (2019). Simple applications of bert for ad hoc document retrieval. *ArXiv*, abs/1903.10972.