Recuperação de Artigos a Partir de uma Modelagem de Perguntas e Respostas

Thiago Coelho

Universidade Federal do Rio de Janeiro

Resumo Este trabalho propõe uma modelagem e um método para utilizar trechos de artigos como consultas para recuperar informações sobre o mesmo assunto tratado pelo artigo em questão. Essa modelagem consiste em considerar estes trechos como perguntas ou respostas para que possa ser aplicado um método proposto no trabalho utilizado como base para este.[4] O método envolve a recuperação de parágrafos que mais estejam em contexto com o trecho utilizado para a consulta.

1 Introdução

Em portais de perguntas e respostas são permitidas várias respostas para cada pergunta, e estas respostas recebem *feedbacks* positivos ou negativos dos usuários, sendo possível, assim, ranquear as melhores respostas baseado nos *feedbacks*. Além disso, normalmente, uma resposta ser eleita como a "melhor resposta", baseado no *feedback* do autor da pergunta ou dos outros usuários.

O artigo *Sanity Check: A Strong Alignment and Information Retrieval Baseline for Question Answering*[4] apresentou uma forma fácil, simples e não supervisionada de identificar a resposta que mais está em contexto com a pergunta. Para isso, é calculado um alinhamento positivo e um negativo entre as palavras da pergunta e das respostas representando cada palavra como um vetor e calculando a similaridade de cosseno entre eles. Também é calculado o idf da pergunta. Utilizando estas informações, ranqueia-se as respostas.

A proposta deste artigo é trazer essa ideia aplicada a outro campo. O objetivo será recuperar trechos de artigos utilizando um trecho solto qualquer para a consulta. O problema será modelado como se este artigo solto fosse uma pergunta e os outros da base de dados fossem respostas.

Dessa forma, havendo bons resultados, será possível considerar a utilidade deste método para coletar mais informações sobre determinado assunto utilizando como consulta dados informativos sobre este mesmo assunto.

2 Ferramentas e Tecnologias

2.1 Natural Language Toolkit

Foi utilizada a biblioteca do NLTK[2] no python para realizar o pré-processamento das palavras, como tokenização e remoção de *stopwords*.

2.2 GloVe: Global Vectors for Word Representation

GloVe[3] é um algoritmo de aprendizado de máquina utilizado para representar palavras como vetores. O treinamento é feito avaliando a coocorrência das palavras dentro de determinado contexto. A partir destes vetores, é possível estimar o quão parecidas duas palavras são semanticamente.

Para a implementação discutida neste artigo, foi utilizada uma base já treinada de vetores de 300 dimensões e vocabulário com tamanho próximo a 1.9 milhão de termos *uncased*. O *download* desses vetores pré-treinados está disponível no site do algorimo.[1]

3 O Problema

Foram escolhidos 21 artigos da Wikipedia em inglês e selecionados, em média, 6 a 7 parágrafos de cada um deles. Para cada artigo, seu primeiro parágrafo foi considerado como uma pergunta e os demais como respostas. Estes parágrafos serão chamados simplesmente de **pergunta** e **resposta**. Todas as perguntas e respostas foram separadas e colocadas individualmente em arquivos diferentes.

Assim como a proposta do artigo *Sanity Check*, a intenção deste algoritmo será definir quais são as respostas que mais estão em contexto com a pergunta. De modo geral, para cada pergunta, deseja-se recuperar as respostas que estavam no mesmo artigo dela.

Como estamos partindo do pressuposto de que não há informação sobre qual artigo possuía cada parágrafo, todas as respostas devem ser candidatas para todas as perguntas.

4 O Algoritmo

O algoritmo implementado se propõe a resolver o problema apresentado acima. Ele receberá como entrada todos os parágrafos selecionados dos artigos, sendo indicados quais são perguntas e quais são respostas, a base de vetores do *GloVe* já treinados e três hiperparâmetros. Sua saída consistirá na ordenação das melhores respostas para cada pergunta, assim como a pontuação de cada uma.

Para facilitar a verificação da eficácia deste método, também foi guardado em qual artigo cada pergunta e resposta se encontram. Isso não foi utilizado em nenhum cálculo de ranqueamento, apenas na avaliação da recuperação.

4.1 Hiperparâmetros

Devem ser definidos, previamente, apenas três hiperparâmetros: K^+, K^- e λ, que são, respectivamente, a quantidade de termos com similaridade positiva, negativa e o peso dado aos termos com similaridade negativa. Eles ficarão mais claros nos próximos tópicos.

Os valores para K^+, K^- e λ utilizados na aplicação do método foram, respectivamente, 5, 2 e 0.5.

4.2 Pré-processamento

Todos os parágrafos, incluindo perguntas e respostas, foram pré-processados. Foi feita a tokenização e remoção de *stopwords* utilizando o NLTK. Também foram removidos todos os caracteres que não fossem letras (números, sinais de pontuação, aspas, parênteses, etc). Por fim, todas as palavras foram colocadas em caixa-baixa.

4.3 *Inverse Document Frequency (idf)*

Para cada termo q_i que ocorre em alguma pergunta, foi calculado o seu *idf* da seguinte forma:

$$idf(q_i) = \log \frac{N - docfreq(q_i) + 0.5}{docfreq(q_i) + 0.5}$$

Em que N é o número de perguntas e $docfreq(q_i)$ é o número de perguntas que contêm q_i.

4.4 Ranqueamento das Respostas

Para calcular a pontuação de cada resposta em relação a uma pergunta, devemos calcular o alinhamento dessa resposta com cada termo q_i da pergunta e somar todos os alinhamentos, cada um multiplicado por $idf(q_i)$. Dessa forma, sendo s a pontuação de uma resposta A relativa a uma pergunta Q, a fórmula é:

$$s(Q, A) = \sum_{i=1}^{T_Q} idf(q_i) \cdot align(q_i, A)$$

T_Q é o número de termos da pergunta Q.

Por fim, o ranqueamento das respostas será feito ordenando-as por esta pontuação.

Alinhamento (*align*) Para calcular o alinhamento de uma resposta com um termo da pergunta (a função *align* utilizada no cálculo de $s(Q, A)$), deve-se calcular os alinhamentos positivos e negativos. Então, a fórmula para o cálculo do alinhamento será:

$$align(q_i, A) = pos(q_i, A) + \lambda \cdot neg(q_i, A)$$

Nessa fórmula aparece o hiperparâmetro λ. Sua utilidade é ponderar o quanto o alinhamento negativo influenciará no valor do alinhamento geral.

Alinhamento Positivo Para selecionar os termos que participarão do alinhamento positivo, é feita uma comparação de todos os termos da resposta com q_i. Para isso, foram utilizados os vetores da base pré-treinada do *GloVe* que representam cada um destes termos. A comparação entre duas palavras foi feita calculando o cosseno entre seus dois vetores. Quanto maior o valor do cosseno, mais similares foram consideradas as palavras.

A partir disso, foram selecionados os K^+ termos de A mais similares a q_i, denominados $\{a^+_{q_i,1}, a^+_{q_i,2}, ..., a^+_{q_i,K^+}\}$.

Por fim, a parte positiva do alinhamento foi calculada da seguinte forma:

$$pos(q_i, A) = \sum_{k=1}^{K^+} \frac{1}{k} \cdot a^+_{q_i,k}$$

Alinhamento Negativo Com o objetivo de penalizar respostas que possuam termos muito fora de contexto, foi considerado também uma parte negativa para o alinhamento. De forma análoga à positiva, o cálculo será feito com base nos K^- termos menos similares a q_i, denominados $\{a^-_{q_i,1}, a^-_{q_i,2}, ..., a^-_{q_i,K^-}\}$.

$$neg(q_i, A) = \sum_{k=1}^{K^-} \frac{1}{k} \cdot a^-_{q_i,k}$$

Os valores numéricos de $a^+_{q_i,j}$ e $a^-_{q_i,j}$ utilizados na conta correspondem aos valores dos cossenos destes termos com q_i.

5 Complexidade

Para calcular o alinhamento ($align(q_i, A)$) foi necessário ordenar os termos da resposta A de acordo com sua similaridade com q_i. Após isso, foram somados os K^+ termos mais similares e os K^- menos similares. Dessa forma, a complexidade da função $align(q_i, A)$ será de $O(T_A \cdot \log T_A + K^+ + K^-)$, sendo T_A o número de termos em um documento A. Como $K^+ + K^-$ deve ser menor que T_A, a complexidade da função $align(q_i, A)$ de fato será $O(T_A \cdot \log T_A)$.

A função $s(Q, A)$ é calculada com um somatório de T_Q valores. Cada valor inclui um cálculo de $align$, logo a complexidade de calcular $s(Q, A)$ será de $O(T_Q \cdot T_A \cdot \log T_A)$.

Calculando o ranqueamento de todas as respostas para um determinada pergunta, a função $s(Q, A_i)$ deverá ser aplicada para toda resposta A_i existente. Considerando, para simplificar o cálculo, que todas as respostas possuem a mesma quantidade de termos e que essa quantidade é T_A (a quantidade média de termos de uma resposta), a complexidade de ranquear as respostas para uma pergunta será de $O(N_A \cdot T_Q \cdot T_A \cdot \log T_A)$, sendo N_A a quantidade de respostas existentes. A complexidade de ordenar as respostas após calcular todas as pontuações é irrelevante à complexidade total do ranqueamento.

5.1 Comparação de Desempenho com a Aplicação do *Sanity Check*

Na aplicação do *Sanity Check*, as respostas para uma pergunta são apenas as que estão no mesmo escopo dela, ou seja, foram consideradas apenas as respostas que os usuários deram àquela pergunta específica. Isso limita consideravelmente a média de N_A, pois normalmente a quantidade de respostas não vai crescer tanto.

Por outro lado, na modelagem proposta neste artigo, as respostas possíveis serão todos os parágrafos existentes na base de dados. Dessa forma, o valor de N_A cresce de forma ilimitada. Quanto mais parágrafos são adicionados à base de dados, maior a quantidade de respostas possíveis para uma pergunta qualquer.

Como a complexidade do ranqueamento para uma pergunta qualquer é de $O(N_A \cdot T_Q \cdot T_A \cdot \log T_A)$, pode-se perceber que essa discrepância afetará negativamente o desempenho em tempo do método proposto.

6 Resultados

O código implementado gerou um ranqueamento das respostas para cada pergunta. No total foram 21 perguntas e 139 respostas vindas de 21 artigos diferentes. Como a base de dados foi pequena, alguns artigos com temas semelhantes foram escolhidos para induzir uma possibilidade de erro maior. Por exemplo, artigos sobre "Grécia Antiga", "Egito Antigo"e "Cleópatra", assim como artigos sobre "Gato", "Cachorro", "Mamba-negra"e "Espécies".

6.1 Algumas Estatísticas

100% dos ranqueamentos retornaram, na primeira posição, uma resposta que estava no mesmo artigo da pergunta. Além disso, 95,24% (20 de 21) também retornaram uma resposta relevante na segunda posição.

6.2 *Mean Average Precision*

Foram avaliados resultados que retornaram as K primeiras respostas para cada pergunta para todo $0 \leq K \leq 139$, sendo 139 a quantidade máxima de respostas possíveis de serem retornadas por pergunta. Para cada um destes testes, foi calculada a *Mean Average Precision* (*MAP*) de todas as consultas.

A *MAP* para o caso em que 139 respostas foram retornadas foi de 87,38%. Entretanto, a *MAP* para apenas 24 respostas retornadas foi de 87,18%, e para 15 respostas foi de 85,79%. Isso indica que os documentos relevantes ficaram bastante concentrados nos primeiros resultados retornados, apontando qualidade ao resultado do método aplicado.

Este resultado indica que seria possível retornar, com quase nenhuma perda de qualidade, uma quantidade bem reduzida de respostas. Isso é essencial para que o usuário possa verificar boa parte dos documentos retornados.

Os gráficos abaixo mostram essa estabilização da *MAP* por volta do ponto com 15 respostas retornadas por pergunta.

Em média houve cerca de 6,62 respostas relevantes para cada artigo.

Figura 1. Gráfico da *Mean Average Precision* por quantidade de respostas retornadas. De 0 a 139 respostas.

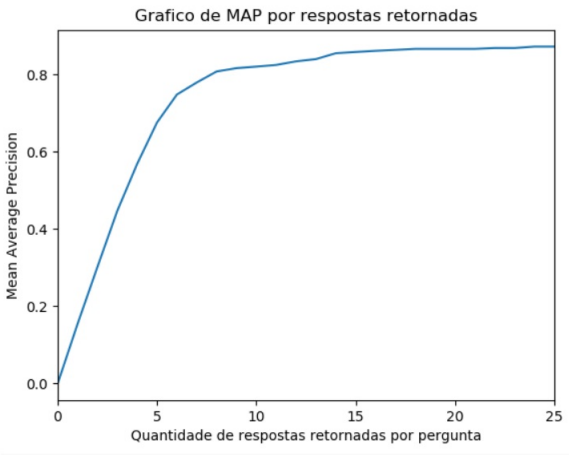

Figura 2. Gráfico da *Mean Average Precision* por quantidade de respostas retornadas. De 0 a 25 respostas.

7 Relevância Real das Respostas

As respostas considerada relevantes foram as que estavam no mesmo artigo da pergunta. Isso foi feito porque o objeto de estudo deste artigo é a viabilidade de descobrir novos trechos de artigos que tenham o mesmo assunto principal. Entretanto, é válida uma reflexão sobre a relevância real de uma resposta para uma pergunta.

Nem sempre é desejado que os trechos retornados tenham exatamente o mesmo assunto principal. Por exemplo, dentre os artigos escolhidos para a base de dados, havia um sobre "Games"e outro sobre "Knight Lore", que é um jogo eletrônico. Um trecho sobre "Knight Lore"é considerado não-relevante para uma pergunta sobre "Games", mas pode ser relevante dependendo da intenção do usuário, pois ainda é uma informação a respeito de jogos.

Além disso, considerando uma quantidade maior de artigos, é possível que haja trechos de documentos diferentes que tenham o mesmo assunto principal. Embora estes trechos sejam relevantes para o usuário, eles são considerados como não relevantes na modelagem feita.

Todavia, isso não foi considerado porque foge do escopo do artigo e tornaria muito complexa a avaliação da relevância de cada resposta para uma pergunta.

8 Conclusão

A modelagem proposta neste artigo para a implementação do método apresentou satisfatória qualidade dos resultados. Devido a estabilização da *Mean Average Precision* a partir de 15 respostas retornadas, pode-se concluir que o número mínimo de respostas retornadas por consulta para garantir uma boa qualidade foi relativamente baixo.

Em contraponto à qualidade dos resultados, a avaliação da complexidade do método aplicado a esta modelagem demonstrou um ponto negativo considerável. Se for aplicado a um sistema onde a base de dados é constantemente atualizada, o seu desempenho será instável e o tempo médio de execução para uma consulta crescerá de forma linear ao aumento da quantidade de dados. Dessa forma, pode ser pouco viável aplicá-lo a um sistema com uma base de dados muito grande e que exija um tempo rápido de resposta a consultas.

Bibliografia

[1] https://nlp.stanford.edu/projects/glove/.

[2] https://www.nltk.org.

[3] Richard Socher e Christopher Manning Jeffrey Pennington. *Glove: Global vectors for word representation.* 2014.

[4] Rebecca Sharp e Mihai Surdeanu Vikas Yadav. *Sanity Check: A Strong Alignment and Information Retrieval Baseline for Question Answering.* SIGIR, 2018.