

Aprimorando o Ranqueamento de Documentos com Grafos de Conhecimento

Nicholas Valle Gross - 122101661

November 2025

1 Objetivo

O objetivo desse projeto é medir o efeito do uso grafos de conhecimento(KG) para expansão de consulta sobre a qualidade de recuperação de documentos quando o mecanismo de base é BM25. Hipótese: a expansão por KG aumenta recall e pode aumentar métricas de ranking (MAP, NDCG), especialmente em consultas com sinônimos/entidades; resultados dependem da qualidade do entity linking e do tipo de expansão. Proposta: comparar a recuperação usando o BM25 e o BM25 + expansão de consulta com KG.

Nota: Ao longo do relatório estarei me referindo a grafos de conhecimento como KG.

2 Fundamentação teórica

Modelos probabilísticos como BM25 ou TF-IDF avaliam a relevância de documentos com base na frequência exata de palavras. O problema dessa abordagem é que as relações semânticas inerentes entre as entidades do documento acabam sendo ignoradas. Então, por exemplo, se a consulta for "capital da França" e um documento falar sobre "Paris", o BM25 não verá a correspondência entre essas entidades apesar de terem alta similaridade semântica. Isso acontece porque esses modelos não entendem o significado das palavras, apenas suas ocorrências.

2.1 Knowledge Graph

Um grafo de conhecimento é uma forma de representar as relações semânticas entre as entidades de um texto por meio de um grafo, no qual seus vértices representam as entidades e as arestas representam as relações semânticas.

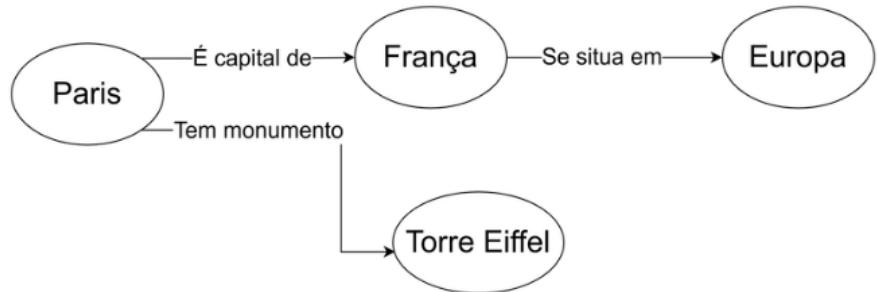


Figure 1: Exemplo de KG

2.2 Expansão de consulta

Expansão de consulta é a reformulação da consulta para obter melhores resultados no processo de recuperação. Para isso, podemos utilizar de:

- Achar sinônimos para palavras
- Achar palavras relacionadas semanticamente
- Adição de novos termos
- Stemming
- Corrigir erros gramaticais

No contexto de KG, a expansão de consulta serve para enriquecer a consulta, englobando termos relacionados e dessa forma recuperar mais documentos relevantes e consequentemente aumentar o recall.

Exemplo de expansão de consulta: “doenças pulmonares” → “doenças pulmonares, doenças respiratórias, bronquite, asma”

2.3 Similaridade semântica

No artigo base[1] é usado o conceito de *relatedness* que é uma métrica para medir o grau de associação entre duas entidades dentro de um KG. Ele mede a proximidade topológica entre duas entidades, ou seja, quanto seus vizinhos (incoming edges) se sobrepõem. Em outras palavras, duas entidades são mais “related” se compartilham conexões com muitas outras entidades semelhantes no grafo.

Originalmente, o *relatedness* era uma métrica usada para avaliar a similaridade entre 2 páginas da wikipédia[2], mais especificamente quantos links externos elas têm em comum.

O *relatedness* de 2 entidades é definido da seguinte forma:

$$R(E_1, E_2) = \frac{\log(\max(|in(E_1)|, |in(E_2)|)) - \log(|in(E_1) \cap in(E_2)|)}{\log(W) - \log(\min(|in(E_1)|, |in(E_2)|))}$$

sendo $in(E_1)$ o conjunto de arestas de entrada de E_1 e W o número total de vértices do KG.

O QDR(Query Document Relatedness) mede justamente o grau de associação entre um documento e a consulta. Compara cada entidade do documento com cada entidade da consulta par a par. É definido como:

$$QDR = \sum_{i=0}^n \frac{\sum_{j=0}^m R(E_{q_i}, E_{d_j})}{m}$$

sendo n o número de entidades da consulta e m o número de entidades do documento.

Porém, quando fui fazer alguns testes usando a fórmula do *relatedness* dada pelo artigo obtive péssimos resultados, com a similaridade indo a 0, ou ficando muito pequena. Isso aconteceu muito provavelmente porque todas as entidades tem o mesmo peso e os grafos ficaram muito esparsos, fazendo com que entidades menos relevantes gerassem ruído no cálculo da similaridade.

Tendo em vista esse problema, resolvi usar outra abordagem. Como algumas entidades são inherentemente mais relevantes que outras em um documento, podemos usar o pagerank para atribuir um peso maior para essas entidades. Com isso, entidades menos relevantes terão um peso menor e vão gerar menos ruído.

Outra escolha que foi feita é desconsiderar o sentido e label das arestas no cálculo da similaridade, para obter melhores resultados. Além disso, no cálculo da similaridade não olhamos apenas para a vizinhança imediata da entidade, mas sim para um grau máximo de separação, que é definido previamente.

Portanto, definimos a similaridade entre 2 entidades como:

$$\text{sim}(e_1, e_2) = \begin{cases} 1 - \frac{d(e_1, e_2)}{D} & \text{se } d(e_1, e_2) \leq D \\ 0 & \text{caso contrario} \end{cases}$$

sendo $d(e_1, e_2)$ a distância mínima entre e_1 e e_2 e D o grau máximo de separação que admitimos.

E a similaridade entre consulta e documento será definida como:

$$\text{Sim}(Q, D) = \frac{\sum_{e_q \in Q} \sum_{e_d \in D} \text{PR}(e_q) \text{PR}(e_d) \text{sim}_{\text{ent}}(e_q, e_d)}{\sum_{e_q \in Q} \sum_{e_d \in D} \text{PR}(e_q) \text{PR}(e_d)}$$

Por fim, iremos calcular o score final para cada documento como a soma entre o seu score do BM25 e a similaridade semântica por KG multiplicada por um fator α .

$$Score(q, d) = BM25(q, d) + \alpha sim(q, d)$$

Escolhi 0.85 para o valor de α , que é o valor tradicional adotado pela comunidade de RI.

3 Implementação

A seguir, vou abordar as etapas que constituem a fase de implementação.

3.1 Construção do KG

A primeira etapa da implementação é justamente construir o KG para cada documento, incluindo a consulta. Essa etapa é dividida em 2 subetapas: extração de entidades e extração de relações semânticas. Para isso, utilizei o spaCy[3], que é uma biblioteca de processamento de linguagem natural(NLP) do python.

3.1.1 Extração de entidades

O nome dessa etapa é auto-explicativo. Precisamos extrair as entidades de um texto. O problema do spaCy é que ele apenas efetua o Named Entity Recognition(NER), ou seja, ele só reconhece entidades nomeadas.

Entidades nomeadas são objetos do mundo real que podem ser denotados por um nome próprio, como pessoas, lugares, organizações, etc. Então, o spaCy reconheceria Rio e Janeiro e Michael Jackson como entidades, mas não reconheceria sapo, por exemplo, já que sapo é um conceito genérico.

Portanto, precisamos dividir a extração de entidades em 3 classes:

- Entidades nomeadas
- Entidades compostas
- Substantivos

Quando o spaCy processa um texto, além de fazer a extração de entidades nomeadas, ele faz a tokenização, e para cada token ele associa atributos como classe gramatical(Part-of-Speech ou POS) e dependência sintática(função da palavra na oração). Esses atributos podem ser usados para fazer a extração de entidades compostas.

A extração de entidades compostas pode ser dividida em 3 tipos: por chunk, por modificador adjetival(substantivo + adjetivo) e por preposição(substantivo + preposição + substantivo). O primeiro pode ser feito usando o método `noun_chunks` que separa o documento em chunks.

No segundo, para cada substantivo, verificamos se algum de seus filhos na árvore de parse são adjetivos(token.dep=amod), se sim extraímos o substantivo e seu modificador como uma entidade.

No terceiro, para cada substantivo, verificamos se algum de seus filhos é um substantivo e sua dependência sintática é de modificador nominal(child_dep=amod), extraímos os filhos dele que são preposições, e por fim concatenamos substantivo, preposição e substantivo.

Além disso, reconheci apenas entidades compostas que tivessem uma página na Wikipédia. Fiz essa escolha, pois muitas vezes o modelo reconhecia entidades compostas que não tem muito valor semântico. Por exemplo, no documento "O rato roeu a roupa do rei de Roma", "roupa do rei" estava sendo reconhecida como uma entidade, apesar de ter pouca relevância semântica dentro do documento.

Finalmente, extraímos os substantivos comuns que não estão nem em entidades nomeadas nem em entidades compostas.

Além disso, fazemos a lemmatização das entidades, que é colocar cada palavra em sua forma canônica, para não gerar ambiguidade.

3.1.2 Extração de relações

Após extrair as entidades do texto, o próximo passo é identificar como essas entidades se relacionam entre si. Essa tarefa é conhecida como extração de relações. O reconhecimento de relações semânticas foi dividido em 2 tipos:

- **SVO:** relações sujeito-verbo-objeto identificadas via dependências sintáticas
- **Relações nominais:** como "X de Y" ou "X com Y", extraídas pela análise das dependências nmod, prep, pobj.

O resultado da extração de relações é um conjunto de tripletas (s-r-o) ou (sujeito-relação-objeto). Então, para cada tripla, pegamos as entidades correspondentes ao sujeito e objeto no KG e criamos uma aresta entre eles com o label r. Apesar do label das arestas estar sendo ignorado nesse trabalho, ele pode ser útil em futuras aplicações.

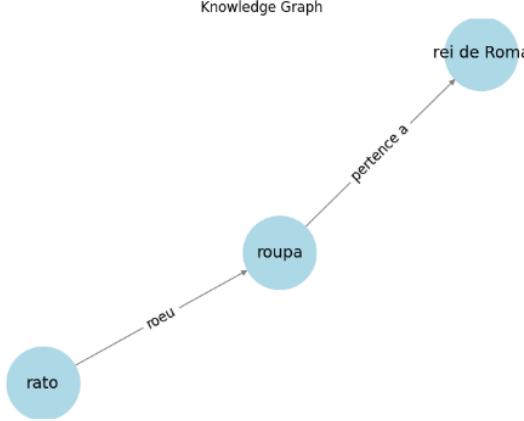


Figure 2: Exemplo de KG construído

3.1.3 Expansão de consulta

Antes de calcularmos a similaridade entre a consulta e os documentos, fazemos a expansão de consulta. Para isso, usei a API da Wikipedia[4] para buscar termos relacionados para cada entidade da consulta. E para cada termo relacionado encontrado, adicionamos uma aresta entre a entidade e o termo com um label genérico.

3.1.4 Similaridade semântica

Com o KG construído e PageRank calculado, computa-se:

1. distâncias entre cada par entidade-consulta × entidade-documento;
2. similaridades ponderadas;
3. média ponderada final.

3.2 BM25

O BM25 foi implementado a partir da fórmula tradicional:

$$BM25(q, d) = \sum_{t \in q} \log \left(\frac{N - n_t + 0.5}{n_t + 0.5} \right) \cdot \frac{(k_1 + 1)f_{t,d}}{k_1(1 - b + b \frac{|d|}{|D|}) + f_{t,d}}$$

com parâmetros padrão $k_1 = 1$ e $b = 0.75$. Além disso, fazemos a normalização do ranking do BM25 para gerar resultados consistentes.

4 Experimentação

Para a etapa da experimentação, escolhi usar novamente a API da Wikipédia para extrair páginas e usá-las como documentos. Fiz essa escolha por questão de praticidade e para ter mais controle sobre o ambiente de experimentação, onde pudesse escolher exatamente os documentos relacionados que iriam para o conjunto de teste.

Com isso em mente, escolhi temas variados, como Saúde, Filosofia, Tecnologia, e para cada tema elaborei uma consulta, escolhi documentos relacionados, e um documento ruído, que não tem nenhuma relação com a consulta, para poder testar a acurácia do modelo. Além disso, manualmente determinei os documentos relevantes para cada consulta. Abaixo está um exemplo de conjunto de teste criado.

```
Ciencia = {"consulta": "O efeito de um buraco negro no espaço-tempo",
           "títulos": ["Horizonte de eventos",
                      "Galáxia",
                      "Relatividade geral",
                      "Guerra Fria",
                      "Espaço-tempo",
                      "Singularidade gravitacional",
                      "Acreção"],
           "relevantes": [0, 2, 4, 5]
}
```

Figure 3: Exemplo de conjunto de teste

Para extrair os documentos, escrevi um script que consulta a Wikipédia, verifica se a página existe e extraí apenas o primeiro parágrafo de sua introdução. Essa decisão é motivada pelo fato de que a introdução geralmente apresenta uma definição concisa e objetiva do termo, contendo informação sem redundâncias ou detalhes excessivos. Esse texto é então limpo e armazenado como um documento, substituindo o campo original de “títulos” por um campo “documentos”. Dessa forma, cada tema passa a possuir um conjunto de textos reais, provenientes de fontes externas, que são posteriormente utilizados em tarefas como similaridade, rankings e construção de grafos de conhecimento.

Por fim, para cada conjunto de teste, executei as 2 abordagens: BM25 e BM25+KG e calculei as métricas de performance: precisão, recall, MAP e nDCG. Para precisão, recall e MAP usei o ranking inteiro e para o nDCG usei os 10 primeiros resultados do ranking. No total foram testadas 20 conjuntos de teste.

5 Validação

A validação do experimento foi feita por meio de:

- Comparação das 2 abordagens(BM25 base e BM25 + KG) usando métricas de performance.
- Uso de documentos reais(Wikipédia).
- Mesmo conjunto de documentos para ambas as abordagens.
- Consultas curtas e de domínio variado.

6 Resultados

Foram calculadas a médias das métricas de performance para cada conjunto de teste e esses foram os resultados:

	Precisão	Recall	MAP	nDCG
BM25	0.383	0.729	0.746	0.849
BM25 + KG	0.433	0.784	0.753	0.843
Variação(%)	+13.05	+7.54	+0.94	-0.71

Figure 4: Média das métricas

Os resultados mostraram que:

- A precisão aumentou significativamente. Isso indica que mais documentos recuperados passaram a ser relevantes quando a expansão semântica via KG foi adicionada.
- O aumento no recall confirma a principal hipótese do projeto: a expansão de consulta por KG aumenta o conjunto de documentos relevantes recuperados, trazendo para o ranking documentos que o BM25 puro não identificaria, especialmente quando a sinonímia ou a variabilidade lexical está presente no domínio da consulta.
- Em alguns casos, documentos relevantes recuperados pela expansão de consulta foram posicionados um pouco mais abaixo do topo, ou documentos irrelevantes ganharam score e ”subiram”, prejudicando a ordenação inicial do BM25, por isso a leve piora no nDCG. Isso significa que o peso da similaridade semântica por KG ainda precisa ser calibrado. Além disso, a expansão de consulta pode ter introduzido ruído, o que influenciou na ordem de ranqueamento dos documentos.

Portanto, com base nos resultados dos experimentos realizados, podemos concluir que expansão de consulta por KG melhora significativamente a capacidade do sistema de recuperar documentos relevantes, embora ainda possa haver espaço para calibrar a influência da similaridade semântica.

7 Trabalhos futuros

Embora o sistema desenvolvido seja capaz de extrair entidades e relações semânticas em língua portuguesa e representá-las em um grafo de conhecimento, ainda existem diversos aspectos que podem ser aprimorados. Como trabalhos futuros, destacam-se:

- Aprimoramento da cobertura semântica das relações. A extração atual de relações baseia-se principalmente em estruturas SVO (Sujeito-Verbo-Objeto) e relações nominais simples mediadas por preposições. Pretende-se expandir o conjunto de regras para contemplar:
 - relações temporais (ex.: “durante”, “antes de”, “após”)
 - relações causais (ex.: “causou”, “resultou em”)
 - relações condicionais e modais
- Incorporação de modelos baseados em aprendizagem de máquina. O pipeline atual depende majoritariamente de heurísticas linguísticas e análise sintática fornecida pelo spaCy. Um avanço seria integrar modelos supervisionados ou transformers (por exemplo, BERTimbau ou LLMs especializados) para:
 - classificar tipos de relações com maior precisão;
 - detectar relações implícitas ou anafóricas;
 - reduzir erro em sentenças com estrutura gramatical incompleta.
- Melhoria do tratamento de co-referência e entidades compostas. Ainda que o sistema implemente resolução básica, há limitações quando:
 - entidades são referidas por pronomes distantes
 - existem múltiplas entidades com nomes semelhantes;
 - há títulos, papéis ou atributos sobrepostos (ex.: “Presidente Lula”, “Governo Federal”).

Propõe-se o uso de métodos mais robustos de resolução de co-referência, bem como o mapeamento semântico entre sinônimos e hiperônimos.

Referências

- [1] Boqi Chen, Kua Chen, Yujing Yang, Afshin Amini, Bharat Saxena, Cecilia Chávez-García, Majid Babaei, Amir Feizpour, and Dániel Varró. Towards improving the explainability of text-based information retrieval with knowledge graphs. *arXiv preprint arXiv:2301.06974*, 2023.
- [2] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with wikipedia pages, 2010.

- [3] Explosion AI. spacy: Industrial-strength natural language processing in python. <https://spacy.io>, 2025. Acessado em: 27-Nov-2025.
- [4] Wikipédia. Wikipédia api. <https://pypi.org/project/Wikipedia-API/>, 2025. Acessado em: 27-Nov-2025.