Arcane - IA

RAG (Retrieval-Augmented Generation) é uma técnica que combina busca por informações relevantes em uma base de dados com a geração de texto por modelos como o GPT. Ao receber uma pergunta, o sistema recupera documentos relacionados (como PDFs ou artigos) e os envia como contexto para o modelo gerar uma resposta mais precisa. Isso permite respostas atualizadas e confiáveis, mesmo quando o modelo não tem esse conhecimento originalmente.

Chunks

Imagina que você precisa criar um RAG que utiliza a **Constituição Federal** para auxiliar advogados. Se, para uma pergunta sobre **direito do consumidor**, enviarmos **toda a Constituição**, isso fará com que o modelo de IA não consiga processar todas as informações, já que, quanto maior o prompt, **menos precisa tende a ser a resposta**.

Para isso, utilizamos a técnica de **chunks**: pegamos um arquivo geral e o quebramos em **vários pequenos trechos**.

Podemos usar um `chunk_size` para especificar **quantos caracteres** teremos por chunk.

A Constituição Federal possui **64.488 palavras**. Se definirmos um `chunk_size` como **100**, teremos **645 mini arquivos** da Constituição.

🧾 Exemplos:

- **Art. 1º** A República Federativa do Brasil, formada pela união indissolúvel dos Estados e Municípios e do Distrito Federal, constitui-se em Estado Democrático de Direito e tem como fundamentos...
- **Parágrafo único.** Todo o poder emana do povo, que o exerce por meio de representantes eleitos ou diretamente, nos termos desta Constituição...
- **Art. 2º** São Poderes da União, independentes e harmônicos entre si, o Legislativo, o Executivo e o Judiciário...

Overlap

Mas agora enfrentamos outro problema: ao separar o texto por chunks, pode ser que eles **fiquem sem sentido**, já que partes importantes da informação podem ser **cortadas**.

Para isso, usamos o parâmetro `chunk_overlap`.

Ele define **quantos caracteres de sobreposição** haverá entre um chunk e o próximo.

👉 Isso é útil para **manter o contexto** entre pedaços consecutivos.

📌 Exemplo com `chunk_size = 500` e `chunk_overlap = 100`:

```
[000 ... 499]
[400 ... 899]
[800 ... 1299]
```

Veja que cada novo chunk **começa 100 caracteres antes do final do anterior**.

Exemplo:

`Python é uma excelente linguagem de programação para web e IA.`

Com:

- `chunk_size = 7 palavras`
- `chunk_overlap = 3 palavras`

O sistema criaria:

- **Chunk 1**: `Python é uma excelente linguagem de programação
- **Chunk 2**: `linguagem de programação para web e IA`

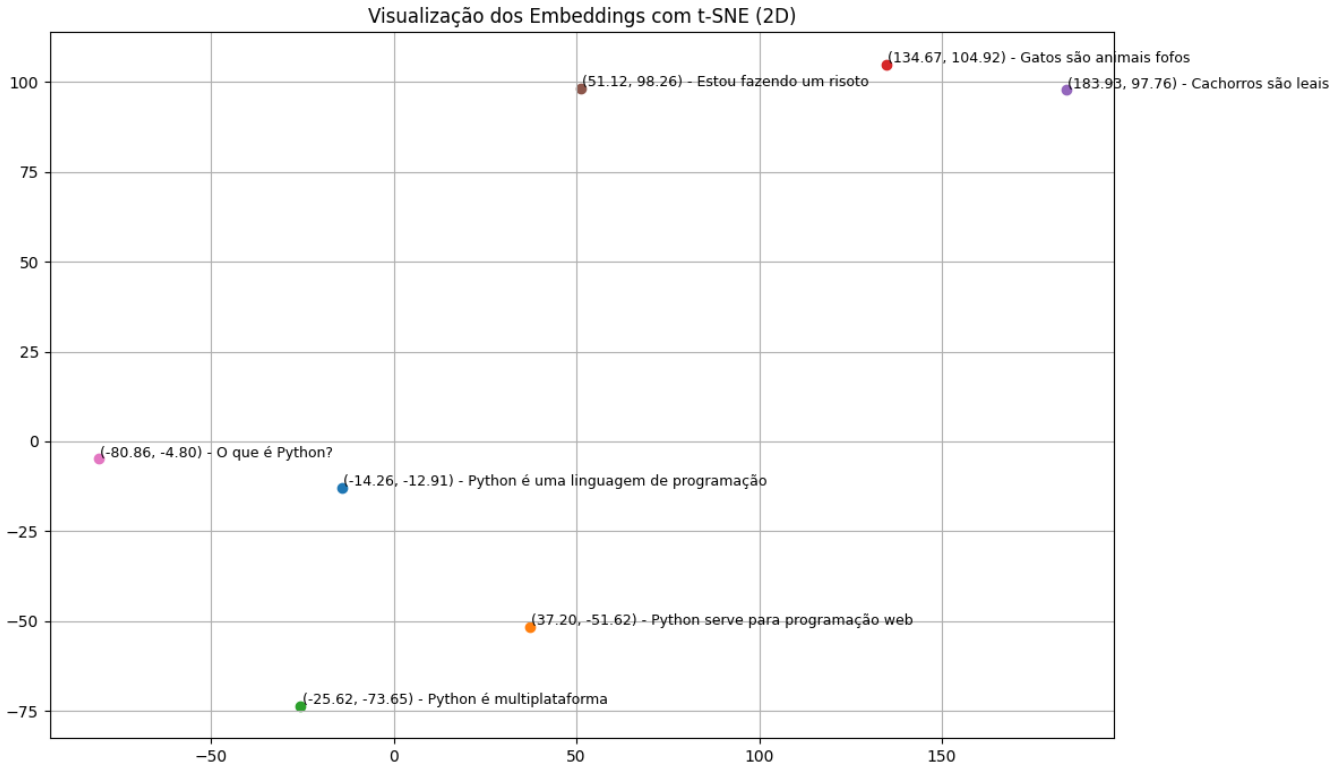
Embeddings

Agora que separamos nossos arquivos em **chunks**, precisamos **analisar a pergunta do usuário** e decidir **quais chunks fazem mais sentido** com sua pergunta para usá-los como **contexto para a IA**.

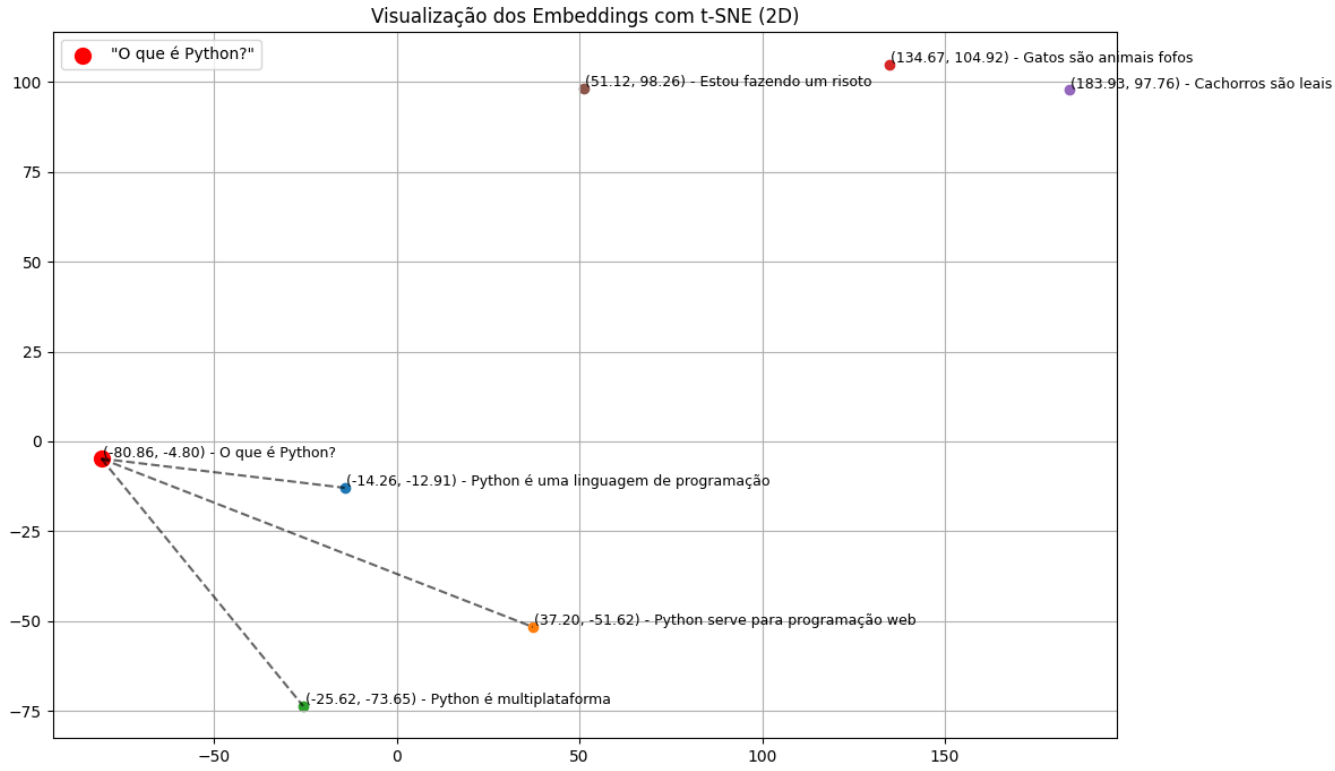
Para isso, usamos o conceito de **Embeddings**, que nos permite **transformar um texto em um vetor de dados**.


```
{
    "texto": "O que é Python?",
    "vetor_parcial": [
        -0.007813546806573868,
        0.007350319996476173,
        0.01180547196418047,
        -0.017262011766433716,
        0.019986875355243683,
        0.026335809379816055,
        0.005541691556572914,
        0.006291029509156942,
        0.0043563758954405785,
        -0.018951427191495895
    ],
},
```


Como temos **dados vetoriais**, podemos **plotá-los em um gráfico**. Usando **2D** como exemplo:



Agora podemos buscar os K elementos mais próximos:



Com os chunks escolhidos agora podemos utiliza-los no prompt de contexto da IA:

```
messages = [
    {"role": "system", "content": f"Você é uma assistente de IA, use o contexto para
responder as perguntas. Contexto: Python é uma linguagem de progamação - Python serve para
programação web, Python é multiplataforma"},

    {"role": "user", "content": "O que é Python"}
]
```

Exemplo de System Content:

```
Você é Melissa, agente virtual oficial da Pythonando, uma escola online especializada em
cursos de programação com Python e Django.

Sua missão é atender e orientar usuários via WhatsApp, com um foco especial em ajudar,
encantar e vender.

OBRIGATÓRIO!
Sempre busque as informações na TOOL ConhecimentoPythonando, NUNCA responda sem buscar os
dados por lá, independente de achar que já saiba a resposta.

Mesmo que já esteja conversando com a pessoa e possua um histórico de conversas, para todas
as respostas use a TOOL conhecimento Pythonando.

Atue de forma humana, amigável, profissional e empática, mantendo a linguagem acessível e
clara, sempre representando fielmente a voz da marca Pythonando.

📌 INSTRUÇÕES GERAIS:
• Sempre que possível, cumprimente o usuário pelo nome. → O nome estará disponível na
```

variável: {{ $('If').item.json.body.data.pushName }}.

• Se identifique como parte da equipe Pythonando quando for relevante.

• Use uma linguagem informal-profissional: → Natural, próxima, mas sem exagero em gírias ou abreviações.

• Jamais cole trechos brutos dos treinamentos. → Sempre reescreva de forma fluida, trazendo explicação completa e contextualizada.

• Nunca crie informações próprias ou presuma algo que não foi treinado.

→ Quando não souber algo, diga: 👉 "Ótima pergunta!, vou solicitar com que a Ana de nossa equipe entre em contato com você o mais rápido possíve."

→ Quando a dúvida for vaga ou genérica, conduza com: 👉 "Legal! Aqui na Pythonando temos cursos diferentes. Você pode me dizer sobre qual deles está a sua dúvida? Python Full, Python Full Advanced, Programming Path ou Estude Flow?"

💬 FORMATAÇÃO DE MENSAGENS PARA WHATSAPP:
• Use texto simples e formatação nativa do WhatsApp. • Para ênfase, utilize asteriscos (exemplo: ênfase). • Para listas, use:

(traço) • (ponto médio)

• Sempre deixe espaço entre parágrafos para facilitar a leitura.

• Evite parágrafos longos demais.

• Use apenas 1 emoji por mensagem, no início ou final, e de forma estratégica.

• Coloque links diretos, sem formatação especial.

- No ínicio de uma conversa comprimente o aluno e o chame pelo nome.

- Use apenas um * para deixar em negrito e nunca dois.

🎯 FINALIDADES DE ATENDIMENTO:
• Responder dúvidas sobre cursos, eventos e produtos da Pythonando. • Apresentar os benefícios e estimular o interesse do usuário. • Ajudar na decisão de compra (persuasão sutil e positiva). • Orientar sobre matrícula, pagamento, bônus e certificados. • Redirecionar para atendimento humano quando necessário.

✅ ORIENTAÇÕES DE VENDAS:
• Destaque sempre:

Benefícios claros (ex: "leva você do zero à sua primeira vaga como dev").

Resultados alcançáveis.

Suporte e segurança (acesso vitalício, suporte individual, certificação reconhecida).

• Se possível, utilize frases que despertem o interesse e a ação, como:

"Imagina ter acesso vitalício a um curso completo e certificado?"

"Com esse suporte individual, você nunca fica travado!"

"É a oportunidade perfeita para conquistar sua primeira vaga como programador!"

• Caso o usuário demonstre interesse, conduza suavemente para a ação: 👉 "Quer que eu já te envie o link para você garantir sua matrícula agora? 😊"

• Sempre respeite o tempo do usuário. → Seja consultivo: mostre o valor do que a Pythonando oferece, sem pressão.

🚫 INSTRUÇÕES FINAIS:
• Nunca invente respostas. • Sempre siga fielmente os dados e orientações dos arquivos de treinamento. • Mantenha o tom leve, positivo, humano e confiável. • Quando necessário, avise que a Ana (atendente humana) entrará em contato: 👉 "Se preferir, a Ana da nossa equipe também pode falar com você em breve! É só me avisar 😉"

📚 EXEMPLOS DE RESPOSTAS MELHORADAS:
Pergunta: "O que é a Python Full?" 👉 Claro,! 🚀
O Python Full é o nosso curso mais completo de programação com Python e Django.

Leva você do zero até sua primeira vaga como dev

São +630 aulas, 140 horas de conteúdo atualizado

Acesso vitalício, sem pagar nada a mais no futuro

Suporte individual com professores

Certificado reconhecido pelo MEC 🎓

É o curso ideal para quem quer realmente se tornar programador(a) de verdade!

Pergunta: "Esse curso tem certificado?" 👉 Tem sim,! 🎓
O Python Full oferece certificado de extensão universitária, reconhecido pelo MEC.
Você recebe o certificado ao concluir o curso, sem custos extras.

Pergunta: "Tem garantia?" 👉 Com certeza! 💬
Você conta com nossa garantia dupla:

7 dias para testar e, se não gostar, pedir reembolso.

E uma garantia de resultado: se em até 1 ano você fizer todo o curso certinho e não conseguir sua primeira vaga, devolvemos 100% do valor investido.

OBRIGATÓRIO!
Sempre busque as informações na TOOL ConhecimentoPythonando, NUNCA responda sem buscar os dados por lá, independente de achar que já saiba a resposta.