**RAG no N8N: Guia Definitivo. Retrieval-Augmented Generation Avançado com Arquivo Jurídico**

**Transcrito por** [**TurboScribe.ai**](https://turboscribe.ai/pt/?ref=docx_export_upsell)**.** [**Atualize para Ilimitado**](https://turboscribe.ai/pt/subscribed?ref=docx_export_upsell) **para remover esta mensagem.**

Então você quer criar uma automação em que um agente inteligente precisa consumir um monte de informação de uma base de conhecimento e você é incapaz de passar essas informações em uma janela de contexto, porque os dados são muitos e ele começa a alucinar. Ou você até já fez um carregamento de embedding das informações, só que seu agente continua não recuperando as informações corretas ou a resposta que você gostaria. Se você está procurando o guia completo de RAG no N8n, esse vídeo aqui é pra você.

O RAG, no seu dia mais feliz, significa Retrieval Augment Generation. Eu gastei meu inglês todo agora, tá? Retrieval Augment Generation significa, pelo contexto do RAG, que Generation é a geração de dados, de informações, onde eu tenho uma capacidade aumentada de fazer a recuperação desses dados externos. Isso significa, na prática, que o RAG é uma recuperação bombada de informações de grande escala.

E por que esse RAG, essa técnica de recuperação em grande escala de informações, é tão importante no contexto de automações no N8n, principalmente na utilização de agentes inteligentes? Geralmente, quando iniciamos as implementações de nossas primeiras soluções de agentes inteligentes no N8n, nós começamos com a edição de dois parâmetros que o Node permite. Esse aqui é o Node da gente, que a gente utiliza para criar a inteligência artificial. Nós associamos, geralmente, com um modelo de LLM, o qual você pode escolher qualquer um e criar sua chave.

E, na sua configuração, a gente precisa passar dois parâmetros. Um é a entrada do usuário, que é aquilo que se pergunta, que é o UserMessage. E o outro é aquele que dita o comportamento, a personalidade desse agente, aquilo que ele deve fazer.

E isso a gente passa, nesse AddOption, no parâmetro SystemMessage. É nesse quadro aqui que a gente vai implementar o comportamento e implementar, passar as informações do contexto do agente. Acontece que toda LLM, todo modelo de linguagem larga, que é essa inteligência artificial, ele possui limitações com a quantidade de entrada dessas informações que ele deve processar por vez.

Essa limitação a gente chama de janela de contexto. No momento desse vídeo, estamos em julho de 2025, e os modelos vigentes, os mais potentes até o momento, são esses GPT 4.1. E se você entra aqui na documentação dele da OpenAI, a gente tem aqui a informação da sua janela de contexto, que ele aguenta, ele suporta 1.047.576 tokens de contexto. E isso considera-se 32.000 como output, como saída, tokens de saída.

Ou seja, aquilo que pode ser processado na entrada é esse valor menos esse. Porque aqui, nessa janela de contexto, também considera-se o output. Por outro lado, também temos aqui o modelo da Gemini, que é da Google.

E esse 2.5 Pro, ao entrar na sua documentação, a gente também constata aqui a sua limitação, sua janela de contexto. A gente tem aqui de input 1.048.576 tokens de entrada e um output de 65.536 na saída. Quando criamos nossos agentes no N8n, nós devemos levar em consideração o modelo escolhido de processamento das informações e essa janela de contexto, que deve considerar tanto o user message, ou seja, aquilo que o lead, aquilo que o usuário pergunta, e o processamento do seu contexto, da sua personalidade, que vem aqui nessa parte de system message.

Ou seja, a soma do que está aqui nesse quadro, juntamente com a entrada do usuário e a resposta, tem que ser menor do que a janela de contexto da LLM. Com o avanço da tecnologia da LLM, essa janela de contexto vai se ampliando. Ou seja, a quantidade de tokens de entrada e saída, ela se amplia quanto mais poderoso for a LLM.

Na maioria dos agentes SDR simples, essas janelas de contexto são suficientes e você não precisa utilizar uma base de dados externas para utilizar o REG como recuperação dessas informações. O próprio N8n, naquela janela de system and property message, são suficientes para você passar toda a informação e diminuir a chance desse seu agente alucinar. Mas existem alguns casos em que os dados são variáveis no seu negócio e você precisa colocar esse conhecimento em uma base externa e utilizar o REG, porque as informações são grandes, para que o agente possa utilizar essas informações sem estourar a sua janela de contexto, sem utilizar todos os tokens de entrada e na saída da sua resposta.

E para utilizar o REG de forma correta e eficiente, você precisa conhecer as etapas do seu processo. Quando exploramos a documentação do N8n, a gente constata que o N8n utiliza uma biblioteca mundialmente famosa e mundialmente utilizada por Davis para fazer a implementação da interface de LLMs, que é o LangChain. É importante você saber que é o LangChain que está por trás do processamento do N8n, porque é ele quem vai editar essas etapas do REG.

É basicamente 5 etapas que a gente deve seguir para a implementação do REG no N8n utilizando o LangChain. Primeiro a gente tem o load das informações, o carregamento, e geralmente essas informações vêm de arquivos externos. Um PDF, um CSV, ou de um CRM ou qualquer outra fonte de informação que você tenha.

Depois do carregamento da informação, a gente precisa fazer um split, ou seja, dividir o texto ou os binários em pedaços de informações chamados de chunks. Então a gente vem aqui, dá um split nessa informação, e existem técnicas para fazer esse split da maneira que você acredita que seja melhor. Depois desse split, a gente tem a filosofia do dividir para conquistar, que aborda todas as áreas da informática.

E depois dessa divisão, a gente tem um processo que a gente chama de embedding, que é transformar esse texto em vetores em um espaço multidimensional. Através desses vetores, a gente vai e faz o armazenamento num vectorstock, que são bases de dados que suportam a estrutura de dados de vetor. E nesse caso, a gente pode utilizar alguns famosos aí do mercado, como o superbase, o postgres como um pgvector, uma extensão, o pinecont, o quadrant, e qualquer outro que venha a surgir, ou que já está surgindo agora como solução para vectorstore.

E por fim, a gente tem a etapa de retrieval, que é o cálculo da distância dos vetores da base de conhecimento relacionados com o vetor da pergunta. Agora complicou, não foi, fraudinha? Eu vou te explicar melhor. Lembra quando você estava lá no ensino médio, que você sentava no fundo e ficava jogando bolinha de papel no professor de matemática, porque ele estava tentando te ensinar o que era um plano cartesiano, e você ficava debochando, dizendo Professor, eu não quero aprender isso não, porque eu nunca vou utilizar isso na minha vida.

Você lembra disso? Lembra? Faça um favor pra mim. Se seu professor de matemática ou professora estiver vivo, ligue pra ele, não mande mensagem não, ligue. Ligue agradecendo, professor ou professora, muito obrigado pela paciência que você teve, e eu ficava dizendo que não ia precisar disso que você estava ensinando.

Chegou o momento de você brilhar. E se esse professor ou professora nesse momento estiver desencarnado, eu desejo cegamente que essa noite, após você assistir esse vídeo, ele venha puxar seu pé enquanto você estiver dormindo. Relembrando sua aula de matemática, que você prestou atenção, mas deve estar um pouquinho esquecido, esse aqui é um plano cartesiano, em que eu tenho as ordenadas y e a bx e a x. E essa reta, elas são escaladas, elas têm uma numeração que vai incrementando, tem o eixo zero aqui, daqui pra cá o valor é negativo, daqui pra cá o valor é negativo, daqui pra cá é positivo e daqui pra cá positivo.

E eu tenho aqui esse ponto, esse ponto A, que ele é representado por essa localização, essa posição 4 e 3. O 4 que está aqui na primeira posição, ele é o valor de x, e esse segundo aqui é o valor de y. Se a gente jogar aqui nesse gráfico, ele vai ter aqui essa posição 4, 4 é o x, e o 3 é o y, então ele vai estar mais ou menos aqui, o que representa esse valor aqui, o 4 e o 3. O que a gente chama de vetor, nada mais é do que um a partir da origem zero, do vértice zero, e encontrando essa posição que é delimitada, que é definida por essa coordenada aqui, 4 e 3. Isso aqui é um vetor. Acontece que pra matemática, o vetor não é apenas uma representação de uma coordenada. É possível também fazer operações matemáticas em cima de vetores.

Então, digamos que eu tenha aqui, por exemplo, um outro vetor, com esses dois vetores aqui, ou mais, é possível fazer operações de adição, de subtração, de raiz, e também o cálculo da distância. Eu posso ter esse aqui, e tudo isso é em operações matemáticas. Nesse caso aqui, nós estamos em um plano 2D, um plano cartesiano, mas podemos escalar isso para planos multidimensionais.

Por exemplo, aproveitando esse mesmo gráfico, a gente pode fazer uma adição, de uma representação de mais um outro plano, que pode ser chamado de plano Z, de eixo Z, na verdade. Em que a gente puxa aqui, criando um plano tridimensional. E aí, a gente vai ter aqui, também, representações de vetores nesse campo.

E no caso, a gente vai ter uma representação desse ponto aqui, com mais um parâmetro, que pode ser tanto positivo como negativo. Por exemplo, isso aqui é mais uma coordenada aqui, mais um valor dentro do nosso plano tridimensional, e que adiciona aqui uma variável Z. Então, nesse caso aqui, o ponto ficaria mais ou menos aqui, nesse espaço. E a gente teria representações de vetores tridimensionais, porque a gente está trabalhando com três planos.

O espaço tridimensional é o máximo que o ser humano consegue representar. Apesar de a gente só conseguir visualizar três planos, isso é uma limitação humana, para a matemática e para a física, a gente consegue trabalhar com mais vetores, fazendo também a redução desses planos. Então, a gente pode ter vetores e posições multidimensionais, ou seja, aquilo que extrapola 2, 3, 4, 5, 6 planos, infinitos planos.

Portanto, quando estamos fazendo um processo de embedding, que é a transformação desse texto em um vetor, a gente está literalmente transformando isso aqui em um vetor multidimensional com vários planos. Aqui a gente vai ter o X, o Y, o Z, e os demais que forem sendo acrescentados, e quem vai definir isso é o LLM. Mas isso não se limita somente a pegar o texto, o caractere, e fazer uma transformação.

O que esse embedding faz é tentar considerar o significado dessas palavras. Não só a sua posição, não só o seu código como letra, mas aquilo que significa no contexto onde está inserido. De forma resumida, o processo de embedding é transformar texto, recuperar seu significado, e dar uma posição dentro de um espaço multidimensional para esse vetor, considerando sua significância.

E é isso que acontece quando a gente faz a carga de informações lá no N8n. A gente vai ter um texto que vai vir de uma fonte externa. Esse texto é transformado em chunks por meio de um split.

Esse split, esses chunks, passam por um processo de embedding. Esse embedding cria, recupera o significado e tenta atribuir um espaço no campo multidimensional. E depois esses chunks, esses vetores, são agregados em um espaço dentro de um vetor, de uma matriz.

E cada um desses vai assumir uma posição dentro do espaço multidimensional. Quando um usuário faz uma pergunta para um agente reg, que está utilizando a base de conhecimento reg, ele vai passar também pelo mesmo processo de embedding. A pergunta chega no nosso agente, ele passa por um processo de embedding, que também é transformado em vetor.

E o que o nosso algoritmo faz é colocar exatamente essa pergunta no meio, também junto com os outros vetores de base de conhecimento. E assim fazer a recuperação baseada na distância. Preste atenção nisso, porque esse é o coração do reg.

Entendendo isso aqui, você entende tudo. Perceba o seguinte, se toda a minha base de conhecimento anteriormente foi transformada em vetores, em um espaço multidimensional, considerando seu significado, quando a pergunta é feita, que também é transformada em vetor, e colocada em um espaço, consequentemente, os vetores do qual eles se aproximam, vão ter significado semelhante. Eles vão ter alguma correlação com aquilo que foi perguntado.

Então, se a pergunta caiu nesse espaço, os vetores que estão próximos aqui, eles têm alguma relação com aquilo que está sendo dito. Então, se eu estou construindo um agente que a base de conhecimento dele é de direito, e eu faço uma pergunta sobre jurisprudência, essa pergunta transforma-se em um vetor, e o nosso algoritmo aqui, ele vai pegar os mais próximos, os vetores mais próximos possíveis, daquilo que foi perguntado. Então, ele transforma isso em chunks.

Ele passa pelo processo aqui, inverso. Antes ele tinha criado um vetor, transformado em um vetor, e agora ele vai ser transformado em texto novamente, fazendo o caminho inverso. E essa informação aqui é passada para o agente, um LLM, e esse LLM utilizando somente esse conhecimento, não toda a base de conhecimento, apenas isso aqui que já foi filtrado, e sabe que tem um significado semelhante, ou está associado de alguma maneira, ele vai produzir uma resposta.

Então, não há necessidade de procurar toda a base de conhecimento, somente aquilo que está relacionado por proximidade. Tá bom, chega de teoria. Como é que eu vejo isso aí no N8n funcionando? Eu já me antecipei aqui no N8n e criei uma pequena automação que faz a recuperação de um arquivo PDF, lá no repositório que eu tenho, que tem o serviço de S3.

Eu poderia fazer isso utilizando o Google Drive, mas como solução mais robusta, eu prefiro utilizar o repositório S3. Se você não sabe o que é um object storage, ele é parecido com o Drive, o Google Drive. Você armazena lá arquivos, música, imagem, aquilo que você quiser.

Acontece que ele utiliza um protocolo S3 e eu pago por essa informação e eu não preciso mais ficar criando credencial para o Google, que sempre tem aquele problema de ficar perdendo conexão. A solução que eu estou utilizando para o object storage é esse Backblaze aqui, que na minha opinião tem me atendido demais nas minhas soluções. Ele tem um preço bem justo, ele cobra 6 dólares por terabyte de informações por mês.

E faz 6 meses que eu estou utilizando isso aqui, ainda não fui cobrado, ainda não cheguei a nem pelo menos acho que 1GB de informação. Então acredito que seja uma ótima sacada para você substituir o Google Drive caso você esteja passando por alguma dificuldade de conexão ou algo do tipo. O sistema aqui é bem simples.

Depois que você faz o seu login, você vem aqui e cria um balde, que é um bucket, eles fizeram a tradução meio errada aqui. E eu vou colocar arquivo-direito-youtube, que é o nome do bucket. Eu vou dizer que ele é público, se você está criando a conta a primeira vez, ele nesse momento vai pedir um cartão de crédito para você, mas não vai cobrar porque você está disponibilizando um bucket público, então ele precisa ali das suas informações financeiras.

Caso você crie um bucket privado, você não vai ser cobrado, mas também sua automação não vai ter acesso. Então eu posso criar um balde aqui, e eu vou dar um upload de um arquivo que eu criei, utilizando vários conceitos jurídicos, várias áreas do direito, para que eu possa tratar melhor o reggae, para que você veja como é que funciona na prática. Esse aqui é o arquivo PDF que eu girei.

Ele é dividido em capítulos, cada capítulo também tem sua especialidade, tem sua área, tem seu tema e um conteúdo de cada tema. Então eu fiz aqui da maneira que cada capítulo aborde uma especialidade, tem direito civil, direito da família, direito penal, direito do consumidor. E é esse arquivo aqui que eu quero carregar, com todas essas nuances, essas características desse arquivo que está relacionado com o direito.

Nesse caso aqui no Backblaze, eu venho aqui em Browse Files, e vou fazer a carga, o upload dessa informação. Eu vou procurar aqui meu bucket que eu criei, nesse caso foi arquivo, direito, YouTube, entro nele, vou fazer o upload, e aqui eu escolho o arquivo que eu vou selecionar dentro dos meus arquivos. Com o meu arquivo carregado, o meu próximo passo é criar um Application Key, para que minha automação tenha acesso a esse bucket.

Eu venho aqui, Add a New Application Key, vou dar o nome para ele de Reg, direito, YouTube, digo que ele vai ter acesso somente a esse bucket, arquivo direito, vou dar aqui a permissão de leitura e escrita, e dou um Create New Key. E aí eu só faço fazer, eu só preciso copiar o Key ID e o Application Key, e criar uma credencial aqui dentro. Fazendo New Credentials, aqui New Credentials, coloco o S-Point, a região, o Access Key e o Secret Key.

Esse Endpoint você consegue encontrar aqui dentro do próprio bucket. Quando você criou o bucket, ele vai dar essa informação aqui, esse é o Endpoint, lembre-se de colocar o HTTPS, aqui, bota aqui o HTTPS, dois pontos, barra, barra, coloca esse Endpoint, a região vai ser essa aqui exatamente, que está na URL, e aqui você coloca o Access Key, o Secret Key, e todo o resto você deixa como está, e dá um Save, e você criou a sua credencial lá no Backblaze, utilizando o S3, bem mais fácil do que utilizar o Google Drive. Depois que eu criei a credencial, Reg, direito, YouTube, que eu criei, passando o Access Key, eu vou pegar uma operação de Get Many, passando o nome do bucket, que é exatamente esse nome aqui, arquivo direito, vem aqui em bucket, foi o bucket que eu criei, que é esse aqui, arquivo direito, YT, passo o nome dele, recupero essa informação, os arquivos que estão lá, que vai ter somente um, faço o download desse arquivo, também utilizando a mesma credencial, e passando o mesmo bucket name, e pego o ID que vem desse node aqui, e aqui nesse passo, eu faço somente a extração, a conversão, eu recupero a informação que foi feita o download aqui, o data, a informação binária, e transformo ele, considerando que ele é um arquivo PDF, em texto.

Se eu fizer a execução desse workflow, eu vou ter essas informações aqui no final, exatamente como está lá no texto. Se eu vir aqui, capítulo 1, direito civil, vou ter todo o texto do PDF, ele vai estar aqui, e é essa informação que eu vou carregar como Reg. Para dar os meus primeiros passos com o Reg aqui no N8n, eu vou utilizar o pgVector do Postgres, porque é um armazenamento local, e eu acho desnecessário ficar utilizando outra solução, se eu não for utilizar todos os recursos, como Superbase, Pinecone, ou Quarantine.

Para mim é o suficiente utilizar a local para esse tipo de solução. Claro que cada desenvolvimento, cada sistema desse de VectorStore, ele tem suas facilidades, seus prós e contras. Para o meu caso aqui específico, para essa aula, e para a maioria dos casos de AGTSDR, eu recomendo utilizar o próprio Postgres com o PgAdmin.

E é você, como gestor de automação profissional, que precisa ter o conhecimento e uma base sólida para poder tomar a decisão de qual solução você vai utilizar quando você for oferecer esse tipo de sistema para o seu cliente. Você pode utilizar o Postgres com a extensão pgVector, você pode utilizar o Superbase, você pode utilizar o Pinecone, o Quarantine, desde que você tenha conhecimento daquilo que vai agregar no seu serviço. É necessário usar Superbase? Você vai utilizar todos os recursos? Faz parte da sua estratégia de negócio? Ok.

Você não tem tantos recursos, você está começando agora, não quer utilizar o Pinecone ou o Quarantine, você vai e faz a instalação do Postgres, cria a instalação do pgVector dentro do seu próprio VPS, não vai ter custo nenhum, vai ser bem mais rápido também, porque ele vai estar na própria VPS sua, vai ter todo aquele processamento interno dos recursos da sua VPS, e já vai estar agregando valor, diluindo seus custos para oferecer essa solução. E também existe a possibilidade de fazer a migração, já que o Superbase, ele é montado em cima do Postgres, então é bem mais fácil você transitar as informações de um para o outro. E nessa minha automação, eu vou adicionar mais um node aqui, eu vou chamar ele de Postgres pgVectorStore.

Vou adicionar um node aqui que é AddDocumentsToVectorStore, cada linha na minha base de dados vai ser considerado um documento, no contexto do Reg, eu vou ter aqui uma conexão com o Postgres, que eu já criei antecipadamente, vou escolher a operação de InsertDocuments, ou seja, inserir essas informações, vou escolher o nome de uma tabela, vou chamar aquele de tblEmbeddings, tblEmbeddings, e vou dar aqui o máximo de batchSize, ou seja, de processamento desses documentos de 200 em 200, isso significa que se eu tiver muita informação, ele vai inserir essas informações no meu banco de 200 em 200 documentos. Não vou adicionar nenhuma opção agora, por enquanto, vou deixar assim dessa maneira, e ele vai escolher aqui, vai me dar permissão para adicionar duas funcionalidades aqui. O que a gente vai fazer é exatamente esse processo.

O que a gente vai fazer é um carregamento do arquivo, fazer um split e fazer o processo de embedding, que é transformar esse texto em vetores, armazenando no VectorStore, que vai ser oposto a isso com o PGAdmin. No nosso caso aqui na nossa automação, ele pede para eu escolher um modelo de embedding, e existem alguns aqui definidos já no N8n. Eu posso ter todas essas opções aqui, você vai escolher uma, eu vou optar aqui pelo embedding, estou na frente, mas é o embedding da OpenAI, eu vou criar aqui uma conexão e escolher um modelo.

Esse embedding aqui já é a conexão que eu tenho com o meu OpenAI, uma chave que eu já fiz antes, e aqui eu tenho três modelos. Esse aqui é um modelo legado, esse aqui é um modelo bom e barato, e esse modelo aqui, ele é mais preciso e mais caro. Esse modelo aqui a gente utiliza quando precisa que o nosso RAG, nossa base de conhecimento, tenha informações precisas.

Ele não pode errar, no caso aqui nosso, o conceito jurídico, por exemplo. Ele não pode ficar alucinando quando se trata de direito, de organização, jurisprudência, também na parte de medicina, de farmácia, ele precisa ser preciso, não existe margem para erro. Eu vou escolher esse large aqui, apesar de ser mais caro, é o que vai atender minha demanda, já que minha área, essa automação que eu estou criando é da parte de direito.

Então eu escolhi aqui o modelo de embedding, e agora eu vou escolher o default data load, aquele que vai fazer o carregamento. Note que eu já escolhi esse embedding aqui, o modelo, agora eu vou pegar o carregamento desse arquivo, como eu posso carregar essa informação. E eu tenho duas opções aqui, um é do GitHub, o algoritmo do GitHub, que é uma solução que recebe e entrega documentos todo dia, então ele é bem ágil esse algoritmo aqui, mas você precisa ter uma credencial e para facilitar aqui a nossa aula, eu vou utilizar o default data load do próprio N8n e aqui eu tenho algumas opções, eu posso dizer que eu vou armazenar dados em JSON ou binário, no caso o meu arquivo está vindo em texto, eu vou armazenar em JSON, eu posso armazenar todos os dados que vão chegar nesse node anterior, é isso que eu quero fazer nesse momento e eu tenho duas opções de text splitting, está vendo que aqui é o local onde eu vou definir o split e eu tenho algumas opções aqui que podem me ajudar eu posso utilizar o simple, que é o default, o padrão dele, que ele vai splitar, ou seja, dividir a cada mil caracteres, dando um overlap de 200 e eu vou explicar isso daqui a pouco, ou customizar isso quando eu clico em custom, da customização ou padronização ele me abre aqui uma outra funcionalidade, que é para eu escolher quem vai ser o splitador do texto, da informação que vai chegar aqui do PDF, se eu clicar aqui, eu vou ter três opções um, caracter text splitter esse é o mais básico e ele define aqui, ele me permite eu colocar um caracter para que eu possa dizer onde é que ele vai quebrar, lembre-se que a gente está dividindo esse texto em chunks, que são pequenos pedaços de texto, então todo o parágrafo que vai chegar aqui na nossa automação, eu vou definir, olha, eu quero que você quebre aqui, por exemplo, ponto final eu quero que você quebre aqui no ponto final e eu quero que caso você não encontre o ponto final, eu quero que você quebre o parágrafo, a frase a cada mil chunks, ou seja, a cada mil pedaços mil caracteres, isso aqui pode ser no meio de uma frase, pode ser no meio de um texto pode ser na frase concreta, desde que ele tenha mil, então a prioridade é encontrar um separador que eu vou definir que no caso aqui eu estou colocando ponto final, e caso ele não encontre ele vai dividir em mil, e esse overlap aqui é a questão do sobrescrita, porque quando eu estou quebrando um texto, eu estou definindo aqui nesse caso o ponto e vírgula ou a cada mil caracteres só que definir quebrar somente mil caracteres pode ser que o agente perca o contexto, ele vai escolher aquela linha da base de dados e não vai saber qual foi, se ele quebrar no meio de uma frase, ele não vai saber qual era o início da frase, por exemplo, digamos que eu coloquei aqui ele não encontrou o ponto e vírgula, é um parágrafo gigante sem ponto, mal escrito, e aí ele veio aqui e quebrou em mil, e quebrou bem no meio de uma frase, e quando o agente pegar esse dado, essa informação, ele não vai saber do que se trata, porque ele só está com a informação pela metade, então a gente coloca um overlap aqui, um valor, para que seja repetido essa informação no início da próxima informação, vou pegar um exemplo aqui do início da frase do texto, para você ter uma ideia de como funciona esse chunk esse overlap, digamos que o algoritmo de chunk que eu escolhi lá, de split, ele veio e quebrou aqui, essa informação ele definiu isso aqui como uma nova quebra aqui, então ele vai colocar isso aqui num array, essa informação, vou deixar a linha aqui para a esquerda, eu vou deixar ele aqui, ele vai quebrar aqui, e ele vai definir isso aqui como um chunk, chunk1 chunk1, e aí ele veio aqui e como eu coloquei um overlap, ele vai pegar essa primeira frase, essas primeiras palavras aqui e colocar, e repetir aqui embaixo e aí ele vai e quebra aqui novamente e aí não há uma perda do contexto porque essas palavras aqui, que é o overlap isso aqui é o overlap, é a repetição no início daquilo que chegou no final do chunk anterior então isso aqui é um chunk2, e assim ele vai repetindo, vou pegar esse último aqui desde o nascimento, vou jogar aqui e ele vai dando overlap, e dessa maneira quando ele pegar, se ele escolher esse chunk2 aqui como conhecimento, no seu algoritmo, transformou isso aqui em vetor, e fez a comparação semântica ele não perde o contexto, porque ele vai saber quais foram as últimas palavras do chunk anterior Voltando para nossos tipos de split, eu tenho aqui o caractere split esse aqui já foi dito, vou excluir esse aqui vou ter o último aqui, que é o token split quando a gente está lidando com o LLM a LLM, as linguagens de inteligência artificial eles não lidam com os caracteres em si eles trabalham com tokens, então toda frase toda palavra, ele se transforma em token, e é esse token que é processado para a LLM, então quando eu utilizo esse texto split de token, o que ele vai fazer é transformar o texto em tokens que vai ser lido pela LLM, e assim eu posso definir o tamanho desses chunks aqui também, chunk size de 1000, e posso dar um overlap de 100 e o outro tipo de splitter que a gente tem, que é o que a gente vai mais utilizar, é o recursivo, caractere recursivo que ele trabalha das duas maneiras, o que ele vai fazer por padrão, ele vai tentar fazer uma quebra em todo fim de frase todo fim de frase ele faz uma quebra, por isso que ele é caractere text splitter, porque ele utiliza esse caractere de quebra de linha, caso ele não encontre a quebra de linha, ele vai tentar dividir pelo chunk size que eu vou definir aqui então ele vai definir pelo tamanho de caractere, a prioridade, quebra de linha caso ele ultrapasse e não encontre ele chega até 1000, aí ele faz a quebra para não deixar o texto tão grande e mais uma vez também, a gente pode colocar aqui um overlap, para que ele faça a sobre descrição dessa informação, e a gente tem aqui as opções de split code, que é o que, dependendo do formato do texto, eu posso definir como é que ele vai considerar os caracteres para fazer a quebra e ele é recursivo, porque ele vai fazer isso sempre nos chunks que ele for tratando, no caso aqui se meu texto fosse em markdown ele ia considerar aquilo que fosse marcado com as tags do markdown, se eu estivesse utilizando o python aqui, que é a entrada, ele ia considerar as quebras de linha do python

**Este arquivo é mais longo que 30 minutos.**

[**Atualize para Ilimitado**](https://turboscribe.ai/pt/subscribed?ref=docx_export_upsell) **em** [**TurboScribe.ai**](https://turboscribe.ai/pt/?ref=docx_export_upsell) **para transcrever arquivos de até 10 horas.**