1. **De que maneira o uso do Word2Vec ajudaria o chatbot a entender diferentes variações das perguntas dos usuários?**

   O Word2Vec é uma técnica de embedding de palavras que converte termos em vetores numéricos, capturando relações semânticas e contextuais entre elas. No contexto de um chatbot, isso permite que o sistema entenda perguntas com variações de palavras ou estruturas, mesmo que nunca tenham sido vistas durante o treinamento.

   O Word2Vec mapeia palavras para um espaço vetorial onde termos com significados semelhantes ficam próximos. Por exemplo:

   Palavras como "cancelar", "interromper" e "parar" terão vetores similares.

   Expressões como "esqueci a senha" e "não consigo acessar minha conta" serão reconhecidas como relacionadas à mesma intenção: recuperação de senha.

   Impacto no chatbot:

   O modelo não depende mais de palavras-chave exatas (ex: "senha") para identificar intenções.

   Perguntas como "Como desativo meu plano?" e "Quero cancelar minha assinatura" são tratadas como equivalentes.

   Com isso, além do chatbot aprender a identificar frases com semânticas semelhantes, ele também se torna mais resiliente a erros de gramática. Tornando a interação com os usuários mais eficiente.

   Plano para utilizar no Chatbot:

   1. Coleta e Pré-processamento de Dados

   Objetivo: Preparar textos relevantes para treinar o modelo Word2Vec.

   fontes de dados: Histórico de conversas do chatbot.

   Ações:

   Limpeza: Remover caracteres especiais, números e stopwords (ex: "o", "de").

   Tokenização: Dividir textos em palavras individuais (ex: "Como redefinir minha senha?" → ["redefinir", "senha"]).

   Normalização: Converter tudo para minúsculas e aplicar lematização (ex: "correndo" → "correr").

2. Treinar o modelo

Objetivo: Criar embeddings que capturem relações semânticas entre palavras.

Ferramentas: Biblioteca gensim (Python) para treinar o modelo.

Opcional: Usar embeddings pré-treinados em português (ex: FastText).

Parâmetros do Treinamento: vector_size: 300 (dimensão dos vetores); window: 5 (palavras vizinhas consideradas); min_count: 5 (ignorar palavras raras).

Saída: Cada palavra é representada por um vetor numérico (ex: "cancelar" → [0.24, -0.57, ..., 0.12]).

3. Integração no Chatbot

Objetivo: Usar os vetores do Word2Vec para identificar intenções dos usuários.

Mapeamento de Intenções: Criar uma lista de intenções (ex: "recuperar_senha", "cancelar_plano"). Para cada intenção, calcular um vetor médio das palavras-chave associadas.

Exemplo:

Intenção: "recuperar_senha".

Palavras-chave: ["senha", "redefinir", "esqueci", "acesso"].

Vetor médio: (vetor("senha") + vetor("redefinir") + ...) / 4.

4. Testar e validar

Objetivo: Fazer casos de teste para validar se o Word2Vec realmente aprimorou o chatbot.

Casos de Teste: Perguntas sinônimas (ex: "Como mudo minha senha?" vs. "Preciso alterar a senha"). Perguntas com erros de digitação (ex: "Esqueci minha senhaa").

Métricas: Acurácia: % de perguntas classificadas corretamente; Cobertura: Quantas variações linguísticas são reconhecidas.

2. **Como o uso de redes neurais recorrentes pode impactar a continuidade do diálogo em uma conversa mais longa com o chatbot?**

As RNNs possuem uma estrutura que permite reter informações de etapas anteriores da conversa. Por exemplo:

Temos o Cenário:

Usuário: "Quero alterar meu endereço de entrega."

Chatbot: "Claro. Qual é o novo endereço?"

Usuário: "Rua Silas Munguba, 123."

Funcionamento da RNN:

A rede lembra que a conversa está no fluxo de "alteração de endereço", evitando respostas desconexas como "Qual pedido você deseja rastrear?". Além disso, Variantes como LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit) resolvem o problema de "esquecimento" em conversas longas:

Mecanismo de Portões:

Esquecer: Decide quais informações do passado são irrelevantes (ex: cumprimentos iniciais). Lembrar: Retém dados críticos (ex: número do pedido em uma conversa sobre entrega).

Exemplo:

Se o usuário menciona "Não recebi meu pedido #123" e depois pergunta "Qual o prazo para reembolso?", a LSTM associa "reembolso" ao pedido #123, mesmo após várias mensagens.

Outro exemplo é pensar no próprio Chat GPT, quando temos uma conversa com ele, mesmo após diversas mensagens ou dias, ele ainda responde de acordo com o contexto da conversa muitas vezes.

Impacto no chatbot:

Agora o Chatbot melhora bastante a experiência do usuário pois se torna capaz de lembrar o contexto das conversas que está tendo, sem precisar que o usuário passe novamente dados já ditos antes.

Plano para utilizar no Chatbot:

1. Objetivo Principal

Garantir que o chatbot: Mantenha contexto durante conversas longas. Entenda dependências temporais entre mensagens. Gere respostas coerentes e relevantes ao histórico da interação.

2. Coleta e Pré-processamento de Dados

Dados Necessários:

Diálogos históricos: Conversas anteriores entre usuários e o chatbot (ex: logs de atendimento).

Estrutura dos Dados:

Pares de entrada-resposta (ex: pergunta do usuário → resposta do chatbot).

Sequências de múltiplas interações (ex: conversas com 5+ mensagens).

3. Treinamento e teste do modelo

4. Integração do modelo ao Chatbot