AVISO DE PROPRIEDADE INTELECTUAL

- ✓ Todo e qualquer conteúdo presente nesse material não deve ser compartilhado, em todo ou em parte, sem prévia autorização escrita por parte do autor.
- ✓ Estão pré-autorizados a manter, copiar e transportar a totalidade desse conteúdo, para fins exclusivos de estudo e controle pessoal, os alunos matriculados em disciplina ou curso de Processamento de Linguagem Natural que tenha sido ministrado em sua totalidade pelo autor, servindo como documento de prova de autorização seu histórico escolar ou declaração da instituição responsável pelo curso, comprovando o referido vínculo.
- ✓ Para o caso de citações de referências extraídas desse material, utilizar:

"CARVALHO, Fabrício Galende Marques de. Notas de aula do curso de Processamento de Linguagem Natural.

São José dos Campos, 2025."

MODELOS DE LINGUAGEM UTILIZADOS EM PIPELINE DE PLN

FABRÍCIO GALENDE MARQUES DE CARVALHO

3.MODELOS DE LINGUAGEM

3.1. INTRODUÇÃO

Esta seção discute o que é um modelo de linguagem e mostra como um determinado modelo pode ser aplicado no contexto de PLN.

São ilustrados os modelos baseados em sacos de palavras e, também, o modelo baseado em *embedding* denominado word2vec.

Para o caso dos modelos utilizando sacos de palavras, será discutido como transformações, tais como a TF-IDF (Term Frequency – Inverse Document Frequency) podem ser utilizados para incorporar importância relativa aos elementos do modelo.

Ainda nessa seção serão estudados conceitos básicos relacionados à similaridade utilizando cosseno e, também, a modelagem de tópicos.

3.2.MODELO MATEMÁTICO DE LINGUAGEM

Um modelo matemático de linguagem, utilizado em PLN, é uma representação numérica usada tanto para dados linguísticos que se devem processar (corpus de entrada) como para dados linguísticos de referência (corpora de texto, corpus classificado, etc.).

Dependendo do tipo de modelo utilizado, um ou vários aspectos relacionados a uma determinada linguagem específica podem ou não ser representados. Ou seja, o modelo utilizado em PLN determina quais aspectos da linguagem natural serão relevantes na construção do sistema.

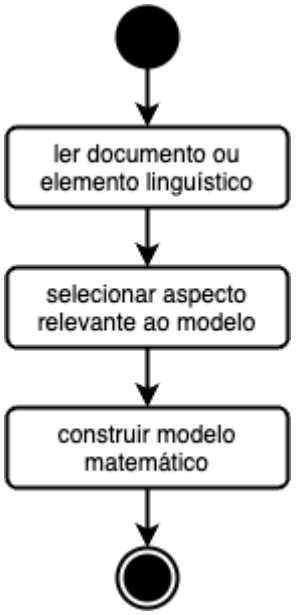
O processo de conversão de informações linguísticas textuais úteis para representação numérica é denominado de extração de características (***feature extraction***).

3.3.EXTRAÇÃO DE CARACTERÍSTICAS DE UM MODELO

Na etapa de extração de características, constrói-se um modelo matemático representativo do documento (***text corpus***).

Após esse processo, um documento ou elemento linguístico passa a ser representado por um vetor numérico que poderá ser utilizado por algum algoritmo de classificação/análise (e.g.: algoritmos baseados em inteligência artificial).

Esquematicamente, o processo de extração de características pode ser representado como:



As entradas para a atividade "ler documento ou elemento linguístico" podem ser, por exemplo, frases, palavras, orações, trechos de frases, etc. Na etapa "selecionar aspecto relevante ao modelo" somente os elementos que afetam o modelo são selecionados para a etapa seguinte de construção, que então gera uma representação, tipicamente vetorial, para o elemento linguístico.

Exemplo: Seja a frase "eu quero estudar". Considerando um modelo de linguagem que mapeie somente verbos de uma frase para um vetor de ocorrências, considerando somente os verbos no infinitivo [querer, poder, estudar], nessa

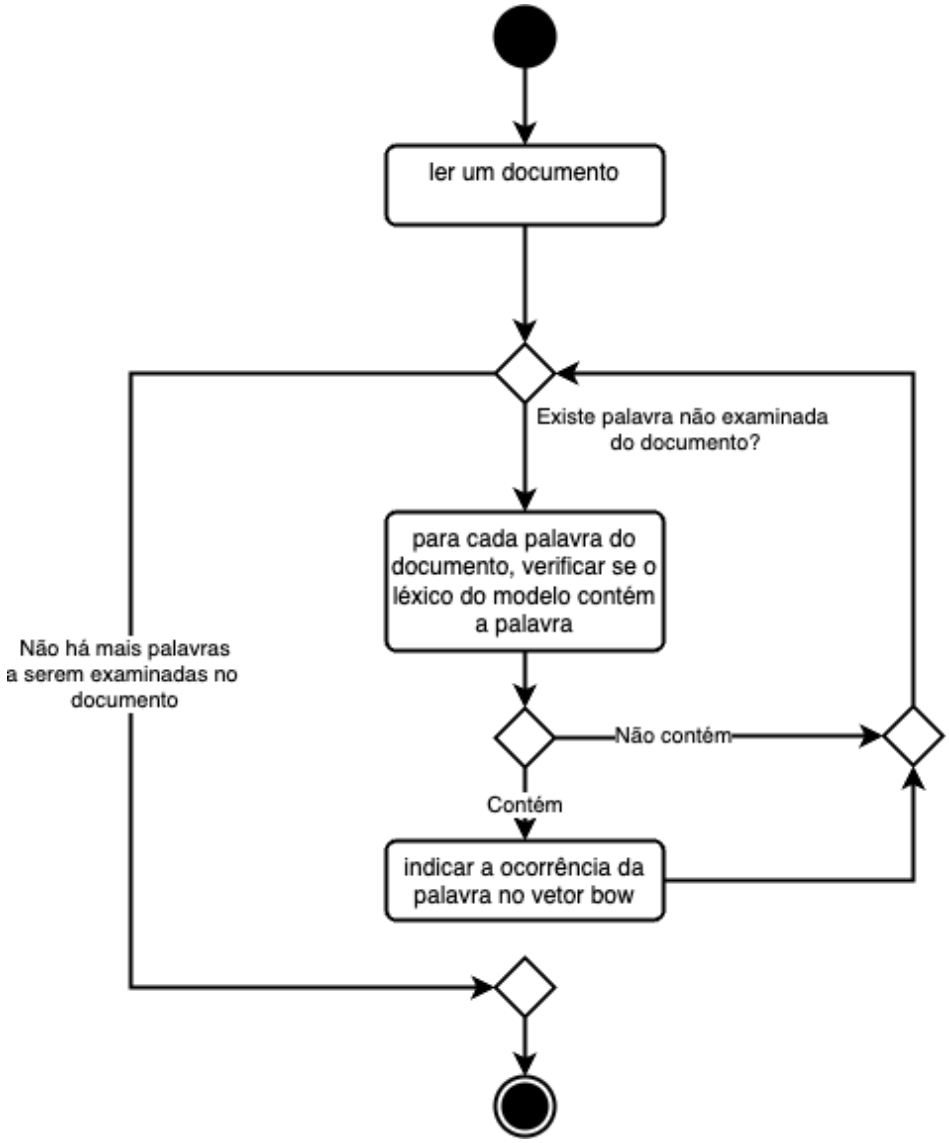
ordem, a saída do modelo, supondo a adequada execução da etapa de pré-processamento, seria: [1, 0, 1].

3.4. MODELO DE SACO DE PALAVRAS

Trata-se de um modelo simplificado que representa um corpus de texto através de um vetor contendo um indicativo da presença ou ausência das palavras presentes no léxico considerado pelo modelo.

Para esse modelo, não importa a ordem nem o relacionamento entre as palavras em uma determinada frase ou oração.

Esquematicamente, o modelo baseado em saco de palavras opera da seguinte forma:



Exemplo: Sejam $\mathbf{T_{in}}$ o texto de entrada, $\mathbf{M_{lex}}$ o léxico do modelo e ***bow*** o bag of words representativo do modelo, então:

$\mathbf{T_{in}}$ = “Gostaria de agendar uma reunião”.

$\mathbf{M_{lex}}$ = [“gostaria”, “poderia”, “seria”, “possível”, “de”, “agendar”, “marcar”, “registrar”, “encontro”, “um”, “uma”, “reuniao’”]

bow = [1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1]

Como vantagens relacionadas a esse modelo de linguagem, podem ser citadas:

- ✓ É um modelo extremamente simples de ser implementado.
- ✓ É de fácil entendimento.

- ✓ Não requer análise gramatical ou processamentos mais complexos.

Já como desvantagens, tem-se:

- × Pode ocorrer a perda de informação contextual, relação de ordem, etc.
- × Sem o adequado pré-processamento, gera entradas e alta cardinalidade.
- × Potencial obtenção de vetores esparsos, podendo gerar dificuldade no treinamento do modelo.

3.5. MODELO DE SACO DE PALAVRAS N-GRAMS

Trata-se de uma generalização do modelo de saco de palavras mas considerando o vetor de características montado a partir de conjuntos de palavras, preservando a ordem em que aparecem em uma frase.

O princípio “por trás do n-gram” está no fato de que palavras que aparecem frequentemente juntas tipicamente se referem a algo com algum significado relevante.

Algumas vantagens desse modelo incluem:

- ✓ Por continuar a ser um saco de conjunto de palavras, preserva simplicidade,
- ✓ É de fácil entendimento.
- ✓ Retém o significado e a estrutura parcial de uma frase/oração/período considerando termos próximos.

Como desvantagens podem ser citadas:

- × O número de combinações de palavras existentes nos n-grams pode se tornar elevado, fazendo com que o vetor de características se torne de elevada dimensão.
- × Potencial obtenção de vetores esparsos, podendo gerar dificuldade no treinamento do modelo.

3.6. MODELO DE SACO DE PALAVRAS COM TRANSFORMAÇÃO TF-IDF

Trata-se de um modelo/transformação que pondera o peso dos termos no vetor de características de acordo com suas ocorrências no documento avaliado e nos corpora tomados como exemplo.

Esse modelo inclui uma manipulação matemática que reduz a importância de termos que são frequentes em muitos documentos ou então que raramente aparecem. Esse fato o torna particularmente interessante para tarefas tais como reconhecimento e classificação de textos em linguagem natural.

Para esse modelo, tem-se que **TF** significa "*term frequency*". O termo TF é o responsável por incorporar a relevância de determinado termo considerando a ocorrência em um documento específico. Já o termo **IDF** significa "*inverse document frequency*" e lida com um determinado termo considerando o contexto mais amplo contendo potencialmente mais de um documento.

Matematicamente, o modelo de transformação TF-IDF é expresso pela seguinte fórmula, considerando uma palavra w_i em um vetor bow = [w_1, w_2, ..., w_i, ..., w_n]

$$w_i = tf_i \ x \ idf_i$$

$$w_i = tf(w_i,d) \ x \ idf(w_i,C)$$

$$w_i = tf(w_i,d) \ x \ \log_{10}\left(\frac{N}{df(w_i)}\right)$$

Na expressão acima, $tf(w_i,d)$ é o número de vezes que o documento w_i aparece no documento *d*, que gerou o vetor *bow, N* é o número total de documentos no corpora de texto do modelo e $df(w_i)$ é o número de documentos, pertencentes ao corpora *C*, em que a palavra w_i aparece.

Como pode ser observado, esse modelo de transformação efetua uma ponderação considerando a ocorrência de uma determinada palavra no contexto isolado de uma frase e no contexto mais amplo envolvendo vários documentos. Um termo que aparece muitas vezes em uma frase possuirá um alto $tf(w_i,d)$. Já um termo que aparece em vários documentos de um mesmo corpora terá um baixo *idf(w,C)*.

Exemplo: Considerando as seguintes frases, as quais serão utilizadas para a obtenção de modelos bow, com transformação TF-IDF:

Frase 1: “gostei, gostei muito do sapato.

Frase 2: “Não gostei do tênis”

Léxico do modelo: [gostei, muito, do, sapato, não, tênis]

bow_1: [2, 1, 1, 1, 0, 0]

bow_2: [1, 0, 1, 0, 1, 1]

Para “gostei”, bow_1: tf_1 x df_1 = 2 x log(2/2) = 0

Para “muito”, bow_1: tf_2 x idf_2 = 1 x log(2/1) = 0,3

Para “do”, bow_1: tf_3 x idf_3 = 1 x log(2/2) = 0

Para “sapato”, bow_1: tf_4 x idf_4 = 1* log(2/1) = 0,3

Para as demais posições em bow_1 (não, tênis), tf = 0, logo tf x idf = 0

Portanto, o bow_1 após a aplicação da transformação TFIDF fica:

bow_{1_tfidf} = [0 , 0.3, 0 , 0.3, 0 , 0]

De modo análogo, para o bow_2, tem-se:

bow_{2_tfidf} = [0 , 0 , 0 , 0 , 0.3, 0.3]

Portanto, conforme pode ser observado, palavras que aparecem em todos os documentos do corpora ou palavras que não aparecem na frase acarretarão componentes nulas no vetor transformado bow_{tfidf}

Um problema que surge ao se utilizar essa transformação será quando os documentos analisados não fizerem parte do corpora original utilizado para compor a transformação. Nesse caso, há duas alternativas possíveis.

a. Recalcular os valores de *idf*(w_i, C'), sendo que C' substitui C agregando-se o documento ao corpora, o que pode ser impraticável sob o ponto de vista prático, em termos de desempenho.
b. Reformular a transformação de modo a se evitar o problema numérico de divisão pro zero, para o caso de termos inexistentes. Nesse caso, a transformação TF-IDF ficaria:

$$w_i = (1 + tf(w_i,d)) \; x \left(1 + \log_{10}\left(\frac{N}{1 + df(w_i)}\right)\right)$$

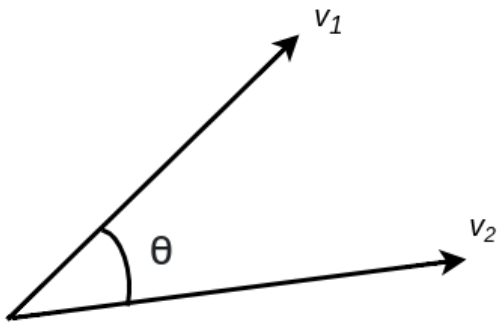
3.7. MODELO DE SIMILARIDADE UTILIZANDO COSSENO

Modelos de similaridade são utilizados para dizer o quanto um determinado documento, pertencente a um corpus, é similar a outro. Esses modelos são utilizados por algoritmos de agrupamento e classificação e tornam possível a utilização de diferentes técnicas de inteligência artificial em tarefas de PLN.

Em geral, modelos de similaridade são modelos que fazem uso de métricas de distância entre os vetores de características que representam um determinado elemento linguístico (e.g., uma palavra, uma frase, etc.).

Um exemplo de modelo de similaridade é o obtido efetuando-se o cálculo do cosseno entre os ângulos formados pelos vetores representativos por cada um dos documentos de um corpus.

Esquematicamente, tem-se



Documento 1, representado pelo vetor v_1, documento 2, representado pelo vetor v_2. Nesse caso, o cosseno representativo da similaridade entre os documentos é dado por:

$$\cos(\theta_{12}) = \frac{v_1.v_2}{|v_1|.|v_2|}$$

Para um corpus de texto contendo N documentos, o resultado do modelo de similaridade 2 a 2 utilizando cosseno será uma matriz simétrica, $M_{similaridade}(NxN)$. Nesse caso, os índices das linhas e das colunas são os responsáveis por indexar os documentos e os valores das células da matriz serão os cossenos (métricas de similaridade).

Uma das principais aplicações dos modelos de similaridade está na utilização de algoritmos de aprendizagem não supervisionada que trabalham através de agrupamento.

Outro exemplo de utilização desse tipo de métrica são os algoritmos de sumarização extrativa. Nesse caso, os documentos que são "mais similares aos demais documentos do corpora" são selecionados como aqueles representativos do sumário extrativo.

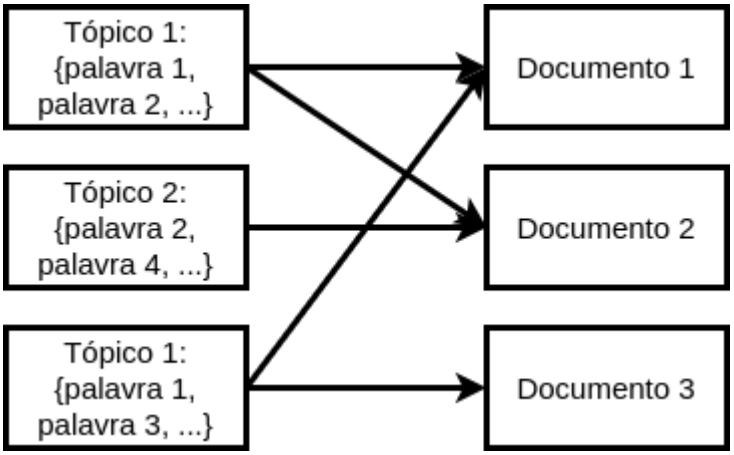
3.8. MODELAGEM DE TÓPICOS

A modelagem de tópicos envolve a extração de temas principais a partir de um corpus de texto.

Um determinado documento, pertencente a um certo corpus de texto, é considerado como possuindo uma mistura de tópicos que são caracterizados por um certo conjunto de palavras.

O modelo de tópico informa o quanto um determinado documento pode ser representado por um certo tópico.

Esquematicamente:



Um modelo popular de tópicos é o LDA (Latent Dirichlet Allocation), que é um modelo probabilístico que informa qual a probabilidade de um determinado documento ser representado por um conjunto contendo N tópicos.

Para um conjunto contendo, por exemplo, 5 documentos e 3 tópicos, a saída desse modelo seria tal como:

tópico 1 | tópico 2 | tópico 3

Documento 0: 0.808529 |0.095344 | 0.096127

Documento 1: 0.105329 |0.095990 | 0.798681

Documento 2: 0.790714 |0.099102 | 0.110184

Documento 3: 0.098138 |0.094206 | 0.807656

Documento 4: 0.093053 |0.813913 | 0.093033

A modelagem por tópicos utilizando LDA é baseada na amostragem de Gibbs. Nesse caso o algoritmo é baseado em probabilidade condicional e funciona da seguinte forma:

1. Para cada palavra em cada uma das frases, atribuir aleatoriamente cada um dos N tópicos T possíveis.

2. Atualize o tópico associado a cada uma das palavras, em cada um dos documentos, considerando a probabilidade condicional de que a palavra pertença ao tópico i, de acordo com a fórmula seguinte:

$$p(T = k \mid w_i, d_j) = \left(nw(T = k, d_j)_{-w_i} + \alpha\right)\left(\frac{nw(T = k, w = w_i)_{-w_i} + \beta}{nw(T = k)_{-w_i} + V.\beta}\right)$$

Na fórmula acima, tem-se:

α e β são os hiperparâmetros de aprendizagem;

nw(T=k, d_j)$_{-wi}$ é o número de vezes que o tópico k aparece no documento d_j, desconsiderando-se a palavra w_i nesse documento;

nw(T=k, w=w_i)$_{-wi}$ é o número de vezes que o tópico k é atribuído à palavra w_i, desconsiderando-se a palavra no documento atual;

nw(T=k)$_{-wi}$ é o número total de palavras atribuídas ao tópico k, exceto a palavra w_i;

V é o tamanho do léxico do modelo.

Uma vez calculadas todas as probabilidades para cada um dos tópicos 1, 2, ..., N, atribui-se à palavra w_i, no documento d_j o tópico T com maior probabilidade condicional calculada.

Por exemplo, supondo as seguintes frases:

Frase 1: quero água

Frase 2: água suja

Teríamos, considerando α e β = 0.1, e supondo-se as seguintes atribuições originais de tópicos (0 e 1) às palavras em cada frase:

Frase 1: {'quero':0, 'água':1}

Frase 2: {'água':0, 'suja': 1}

Considerando a palavra quero (w_1), no documento 1 (d1), tem-se

Para o tópico 1:

$$nw(T=0,d=1)_{w1} = 0$$

$$nw(T=0,w=w1)_{-w1} = 0$$

$$nw(T=0)_{\_w1} = 1$$

Portanto, a fórmula fica:

p(T=0 | d1, w1) = (0+0.1)* (0+0.1)/(1+0.1*3) = 0.1* (0.1)/(1.3) = 0.01/1.3 = 0.007

Para o tópico 2:

$$nw(T=1,d=1)_{-w1} = 1$$

$$nw(T=1, w=w_1)_{-w1} = 0$$

$$nw(T=1)_{-w1} = 2$$

Portanto, a fórmula fica:

$p(T=1| d_1, w_1) = (1+0.1)*(0+0.1)/(2+3*0.1) = 1.1*0.1/2.3 = 0.11/2.3 = 0.0478$

Como $p(T=1|d_1,w_1) > p(T=0|d_1,w_1)$, a palavra quero, terá atribuído o **tópico 1**.

Se tivéssemos começado com a palavra w2 = água, teríamos:

Para o tópico 0:

$$nw(T=0,d=1)_{w1} = 1$$

$$nw(T=0,w=w1)_{-w1} = 1$$

$$nw(T=0)_{\_w1} = 2$$

$$p(T=0|d_1,w_1) = (1+0.1)(1+0.1)/(2+0.3) = 1.21/2.3 = 0.526$$

De modo análogo para T =1, teríamos

$$nw(T=1,d=1)_{-w2} = 0$$

$$nw(T=1, w=w_2)_{-w2} = 0$$

$$nw(T=1)_{-w2} = 1$$

Portanto, teríamos:

$$p(T=1|d_1,w_2) = (0+0.1)(0+0.1)/(1+0.3) = 0.01/1.3 = 0.0076$$

Logo, como $p(T=0|d_1,w_2) > p(T=1|d_1,w_2)$, palavra w2 = 'água' terá atribuído o tópico 0.

Esse processo é repetido para todos os documentos e todas as palavras até que não haja mais mudança dos tópicos atribuídos a cada uma delas (convergência do algoritmo.

3.9. MODELAGEM COM EMBEDDING USANDO WORD2VEC

Os modelos baseados em contagens de palavras que foram vistos anteriormente são ditos modelos tradicionais de características.

Esses modelos, apesar de serem fáceis de serem compreendidos, são modelos que tendem a produzir vetores esparsos (contendo muitos zeros) e que não preservam relação de ordem ou similaridade existentes entre palavras em um dado contexto

Considere-se, por exemplo, duas frases que representam a opinião positiva de dois consumidores, referentes a dois produtos:

Frase 1: Gosto muito do produto!

Frase 2: Amo esse produto!

Efetuando-se a conversão para letras minúsculas e montando-se um léxico de modelo formado pelas palavras {'gosto', 'do', 'produto', 'amo', 'esse', 'produto'} tem-se a seguinte modelagem utilizando-se *bag of words*:

doc_1 = [‘gosto do produto’]

doc_2 = [‘amo esse produto’]

bow_1 = [1, 1, 1, 0, 0]

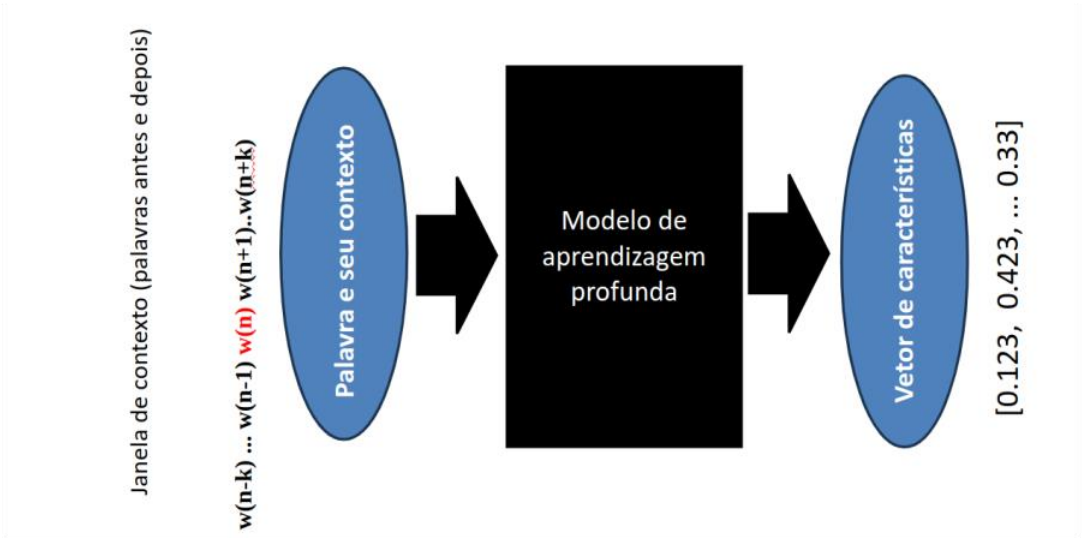
bow_2 = [0, 0, 1, 1, 1]

Como pode ser observado, os *bag of words* representativos de documentos muito similares são diferentes e incluem 40% de zeros (informação não relevante) no vetor de características. Esse aspecto torna a utilização dessa representação problemática para tarefas tais como classificação e reconhecimento de padrões.

Pensando em resolver o problema de elevada dimensionalidade e de vetores esparsos dos modelos tradicionais, uma equipe do Google propôs, por volta de

2013, um modelo que gera um vetor pouco esparso e denso para cada uma das palavras pertencentes a um corpus.

Esses modelos, conhecidos como ***"word embeddings"*** capturam tanto a relação de ordem como o contexto relacionado a uma dada palavra.



No esquema da figura anterior uma palavra na posição n, representada por w(n), é modelada tendo-se como base o seu contexto, representado por uma janela de tamanho k. A janela utiliza então todas as palavras ao redor de w(n) que estão 1, 2, ..., k posições distantes dessa. O sentido em se utilizar tal modelo parte do pressuposto de que palavras com sentidos similares serão utilizadas próximas das mesmas palavras, tal como mostrado no exemplo relacionado à opinião positiva relacionada a um produto (i.e., as palavras amo e gosto estão próximas da palavra produto).

Além da questão contextual, a utilização desse tipo de modelagem permite ao desenvolvedor especificar qual o tamanho do vetor representativo de uma determinada palavra ou frase. Dessa forma os vetores serão não esparsos, ou seja, incluirão menos zeros, e possuirão representações similares para palavras que possuem significados similares em um determinado contexto.

Para o caso de frases, a modelagem utilizando ***embeddings***, obtém o vetor representativo de uma determinada frase considerando a média de seus embeddings representativos de cada uma das palavras.

Suponhamos, por exemplo, que as palavras 'eu' e 'gosto' possuam embeddings correspondentes iguais a [0.5, 0.3, 0.7] e [0.7, 0.1, 0.1]. Então, o embedding correspondende à frase 'eu gosto!' Será:

$$[(0.5+0.7)/2 , (0.3+0.1)/2, (0.7+0.1)/2] = [0.6, 0.2, 0.4]$$

Considerando $wv(w_i)$ como sendo a função word2vec correspondente à palavra w_i, então para uma frase *f* contendo palavras w_1, w_2, ..., w_n, a representação word2vec(f) = *wv(f)* será dada por:

$$wv(f) = \frac{\sum_{i=1}^{n}\left(wv(w_i)\right)}{n}$$

EXERCÍCIOS E PROBLEMAS:

TERMINOLOGIA E CONCEITOS

TC.3.1. Qual tipo de problema pode surgir na montagem de um modelo do tipo *bag of words* caso etapas tais como a remoção de caracteres especiais (ex. sinais de pontuação), stemização, lematização e conversão para minúsculas/maiúsculas não sejam efetuadas? Ilustre isso para as seguintes frases que fazem parte de um mesmo corpus de texto (i.e.: são usadas para a montagem do léxico do modelo):

Frase 1: Eu quero tomar água!

Frase 2: eu, prefiro tomar café.

Frase 3: Eu preferiria tomar café quente.

Frase 4: Eu queria tomar água?

Na sua resposta, ilustre como seriam representados o léxico do modelo e os modelos de frase do tipo bag of words correspondentes. Na sua resposta inclua também qual a cardinalidade do léxico de palavras e dos modelos com e sem os elementos da pipeline de pré-processamento.

TC.3.2. Qual a relação entre as etapas de pré-processamento de texto e a redução de dimensionalidade quando se lida com extração de características? Ilustre isso considerando 3 exemplos que façam uso de stemização e/ou lematização. Ilustre, também, como os processos de stemização e lematização podem interferir na perda do significado de uma frase que será manifestado pela obtenção de vetores *bag of words* idênticos para frases que são semanticamente diferentes.

TC.3.3. Descreva como ficaria o léxico do modelo e o vetor de características *n-gram* para n=1, 2 e 3 para os seguintes documentos pertencentes ao mesmo corpus:

Frase 1: Eu não gostei do produto e o produto parece ruim.

Frase 2: O produto parece bom.

Frase 3: O produto parece ruim.

Frase 4: Parece muito ruim esse produto.

TC.3.4. Para o exercício **TC.3.3,** considerando um modelo *bag of words,* com n=1, mostre como se calcula o valor da transformação TFIDF para as palavras produto e ruim, ambas na frase 1. Utilize as expressões fornecidas no material das aulas de PLN. Adicionalmente, forneça o valor dos vetores bag of words, com transformação TFIDF para todas as frases mostradas no exercício.

TC.3.5. Considere as seguintes frases utilizadas em um sistema de processamento de linguagem natural:

Frase 1: Gostaria de pedir ajuda para alguém.

Frase 2: Gostaria de ajuda.

Frase 3: Gostaria de sair do sistema.

Frase 4: Sair do sistema!

Frase 5: Reiniciar o sistema!

Considerando um modelo do tipo bag of n-words, com n= 1 e n =2, monte o léxico do modelo e o vetor de caraterísticas representativo de cada frase considerando:

a) A mera contabilização dos n-gramas na frase.
b) A aplicação da transformação TFIDF para o vetor de contabilização.

Diga se n=1 ou n=2 afeta o tamanho e o valor das componentes do vetor de características obtido para o modelo. Na sua resposta, especifique os valores dos termos IDF para os unigramas e bigramas. Apresente o detalhamento do cálculo de pelo menos 4 IDFs, sendo 2 para unigramas e 2 para bigramas.

TC.3.6. Ilustre o cálculo efetuado pelo algoritmo LDA, utilizando amostragem de Gibbs para as palavras quero e comer, no documento 1 e bem no documento 2. Suponha que na inicialização do algoritmo, os tópicos 1 e 2 foram distribuídos, inicialmente, tal como mostrado após cada uma das frases abaixo

Frase 1: Eu quero comer. (t=1, t=1, t =0)

Frase 2: Comer bem faz bem. (t=1, t=0, t=1, t=0)

Frase 3: Te quero bem. (t=0, t=1, t=1)

TC.3.7. Ilustre o algoritmo de agrupamento hierárquico (que foi demonstrado em sala de aula e que possui arquivo disponível no repositório do github), considerando as seguintes palavras com os seus respectivos vetores de embeddings:

Palavra 1: gato, embedding: [0.5, 0.5]

Palavra 2: felino, embedding: [0.56, 0.49]

Palavra 3: cão, embedding: [0.1, 0.2]

Palavra 4: lobo, embedding: [0.12, 0.25]

Na sua resposta, faça uma ilustração explicando a quantidade de clusters formados, seus sucessivos agrupamento (utilize o critério botton-up, menor distância) e construa o dendrograma gerado a cada iteração. Para cada rodada, especifique o critério de distância que é utilizado para separar os clusters.

PRÁTICA DE PROGRAMAÇÃO

PP.3.1. Baseando-se nos exemplos fornecidos pelo professor, que fazem uso da biblioteca scikit-learn, ilustre a obtenção de um vetor do tipo *bag of words* com transformação do tipo TFIDF para **três documentos que representem reviews de produtos em um site de e-commerce**. Execute todas as etapas de pré-processamento necessárias para normalizar os dados. Uma vez

obtidos os vetores, faça o cálculo de similaridade utilizando a função cosseno e diga quais reviews são mais similares e quais reviews são mais distintas.

PP.3.2. Baseando-se nos exemplos fornecidos pelo professor, que fazem uso da biblioteca scikit-learn, ilustre a obtenção de um vetor do tipo *bag of words* com transformação do tipo TFIDF para **três documentos que representem solicitações de atendimento que chegam em um service desk**. Execute todas as etapas de pré-processamento necessárias para normalizar os dados. Uma vez obtidos os vetores, faça o cálculo de similaridade utilizando a função cosseno e diga quais solicitações são mais similares e quais solicitações são mais distintas. Ilustre a matriz de similaridade.

PP.3.3. Baseando-se nos exemplos fornecidos pelo professor, que fazem uso da biblioteca scikit-learn, ilustre a obtenção vetores do tipo *embeddings* para **três documentos que representem reviews de produtos em um site de e-commerce**. Execute todas as etapas de pré-processamento necessárias para normalizar os dados. Uma vez obtidos os vetores, faça o cálculo de similaridade utilizando a função cosseno e diga quais reviews são mais similares e quais reviews são mais distintas. Ilustre a matriz de similaridade.

PP.3.4. Baseando-se nos exemplos fornecidos pelo professor, que fazem uso da biblioteca scikit-learn, ilustre a obtenção vetores do tipo *embeddings* para **três documentos que representem olicitações de atendimento que chegam em um service desk**. Execute todas as etapas de pré-processamento necessárias para normalizar os dados. Uma vez obtidos os vetores, faça o cálculo de similaridade utilizando a função cosseno e diga quais solicitações são mais similares e quais solicitações são mais distintas. Ilustre a matriz de similaridade.

PP.3.5. Considerando um corpus de texto contendo revisões de produtos, selecione algumas revisões que possam ser caracterizadas como positivas, negativas ou neutras. Treine seu modelo de aprendizagem de máquina utilizando o K-Nearest Neighbors (KNN), tendo como base o código-fonte fornecido pelo professor, para que seja capaz de classificar uma determinada revisão informada pelo usuário, diferente daquela que foi utilizada no treinamento do modelo. No seu modelo utilize a modelagem usando *bag of words com* transformação TFIDF. Salve os dados do seu modelo treinado em um arquivo pickle, recarregue e demonstre a sua utilização para nova classificação (ou seja, para uma revisão que não tenha sido classificada).

PP. 3.6. Repita o exercício 3.5 mas utilizando um Multilayer Perceptron (MLP).

PP.3.7. Desenvolva um serviço que recebe como entrada um texto enviado em um e-mail para o service desk. O texto pode ser classificado como uma das seguintes categorias:

1. Incidente.
2. Dúvida.
3. Elogio.

Utilizando uma modelagem do tipo *bag of words* e um classificador do tipo Multilayer Perceptron, desenvolva um sistema que, dado um texto de entrada, gera um rótudo que corresponde à classificação do documento. Esse tipo de sistema serve para encaminhar adequadamente a entrada do usuário para o colaborador adequado da equipe do service desk.

PP.3.8. Um sistema de classificação de solicitações geradas para um *service desk* pode ser utilizado para checar o real desempenho de um operador que efetua a classificação manual de cada solicitação injetada no sistema. Utilizando a modelagem do tipo word2vec, desenvolva um sistema que classifica as solicitações geradas por parte de um usuário e compara com as mesmas classificações só que geradas por um operador. Simule o comportamento do sistema considerando o seguinte:

1. Crie 9 entradas para o sistema, das quais 3 representem um incidente, 3 representem uma dúvida e 3 representem um elogio. Treine seu modelo utilizando um Multilayer Perceptron.
2. Crie outras 6 reviews que tenham sido erradamente classificadas por um operador e que recaiam em uma das 3 categorias que foram utilizadas para treinar o seu modelo com as 9 reviews originais.
3. Execute seu modelo treinado sobre as três reviews erroneamente classificadas pelo operador humano e veja se o erro de operação foi detectado (i.e., seu modelo de IA e PLN classifica corretamente enquanto o operador não).

PP.3.9. Demonstre a técnica de agrupamento hierárquico de documentos similares utilizando alguns dados de reviews de produtos. Ilustre e explique o dendrograma em especial no que se refere aos pontos de corte para as distâncias. Faça uso de dados de reviews de produtos e não se esqueça de normalizar os dados antes de efetuar a montagem dos vetores de características.

PP.3.10. Utilizando a técnica de agrupamento hierárquico, agrupe entradas relacionadas a solicitações de um usuário para um service desk. Desenhe o dendrograma e mostre como essa técnica pode ser utilizada para agrupar solicitações similares.

PP.3.11. Ilustre a modelagem de tópicos, com LDA, utilizando alguns documentos representativos de revisões de produtos (mínimo 10 documentos e 3 tópicos). Efetue todas as etapas de pré-processamento adequadas antes de efetuar a modelagem. Diga quais são as palavras relacionadas mais relacionadas a um determinado tópico.

PP.3.12. Ilustre a utilização da modelagem de tópicos, utilizando LDA, considerando alguns documentos representativos de solicitações enviadas a um *service desk*. Utilize no mínimo 10 documentos e três tópicos. Efetue todas as etapas de pré-processamento antes de efetuar sua modelagem (e.g., elimine as stopwords e faça uso de lemmatization ou stemming). Mostre como a modelagem de tópicos pode ajudar em insights para a elaboração de um dashboard de service desk e como isso pode ajudar um gestor a tomar boas decisões de negócio. No seu exemplo, utilize textos de solicitações que sejam favoráveis a observação dos resultados (ex. Utilize palavras típicas em solicitações que podem ser atreladas, por exemplo, a incidentes).

PP.3.13. Desenvolva um chabot, utilizando modelagem bag of words, que pode ser utilizado para auxiliar o usuário a navegar em uma aplicação web. Seu chabot pode ter uma interface via prompt de comando ou web. Ele deve ser capaz de responder a perguntas tais como "Como instalo um programa xyz?". Quando o usuário utiliza uma palavra tal como "sair" ou "quero sair" a aplicação deve ser encerrada ou o chatbot deve se despedir do usuário.

PP.3.14. Desenvolva uma aplicação que efetue uma busca semântica. Sua aplicação deve ter uma interface web e a busca deve funcionar independentemente da entrada do usuário possuir as palavras exatas buscadas. Exemplo: Se o usuário digitar "preciso instalar o windows" o retorno deve listar tópicos tais como: "guia de instalação de sistemas operacionais", "dúvidas sobre

instalação", etc. Na sua modelagem utilize o conceito de matriz de similaridade de cosseno e modelagem word2vec.