Aula 2 – Computação Cognitiva

Pré-processamento

Bag-of-words

- Encontrar palavras chaves;
- Necessidade de tokenização anterior;
- Necessidade de remoção de conectivos , pontuação, artigos e etc;
- Informa a quantidade de ocorrência do termo no texto;

Bag-of-words

- Exemplo:
  - Texto : The cat is in the box. The cat likes the box.

  - Bag-of-words:
    - The: 2
    - cat: 2
    - Is: 1
    - In: 1
    - the: 2
    - box: 2
    - . : 2
    - likes: 1

Exemplos

```
from nltk.tokenize import word_tokenize
from collections import Counter
Counter(word_tokenize("""The cat is in the box. The cat likes the box.
                The box is over the cat."""))
```

```
Counter({'.': 3,
         'The': 3,
         'box': 3,
         'cat': 3,
         'in': 1,
         ...
         'the': 3})
```

```
counter.most_common(2)
```

```
[('The', 3), ('box', 3)]
```

Prática

Questão 1

`code1.py`

contagem de palavras com bag of words

Neste exercício, você construirá um contador de palavras usando um artigo da Wikipedia, que foi pré-carregado como `article`. Tente fazer o bag of words sem olhar o texto completo do artigo e adivinhar qual é o tópico! Se você quiser dar uma olhada no título no final, nós o incluímos como `article_title`. Observe que o texto deste artigo teve muito pouco pré-processamento da entrada bruta do banco de dados da Wikipedia.

Instruções:

1. importe `word_tokenize` da NLTK e `Counter` da lib `collections`.
2. Tokenize `article` com `word_tokenize`
3. Utilize List comprehension com `t` como elemento iterador para converter as letras maiusculas em minúculas. use `.lower()` como método para realizar a conversão.
4. Crie um bag-of-words de `lower_tokens` utilizando `Counter`. Armazene o resultado em `bow_simple`.
5. Utilize o método `.most_common()` para printar os 10 tokens mais comuns deste conjunto

Pré-processamento

- As palavras de maior frequência se tornam as palavras relevantes;

- Auxílio em treinamento de classificadores;

- Exemplos:
  - Tokenização;
  - Lematização;
  - Todas as letras minúsculas;
  - Remoção de caracteres especiais e pontuação.

Exemplos

```
from ntlk.corpus import stopwords
text = """The cat is in the box. The cat likes the box.
                    The box is over the cat."""
tokens = [w for w in word_tokenize(text.lower())
                    if w.isalpha()]
 no_stops = [t for t in tokens
                      if t not in stopwords.words('english')]
Counter(no_stops).most_common(2)
```

```
[('cat', 3), ('box', 3)]
```

Prática

Questão 2

`code2.py`

Pré-processando o texto

Agora, você deve aplicar técnicas para pré-processar o texto e obter melhores resultados. Você precisará remover `stop words` e caracteres não alfabéticos, lematizar e executar um novo conjunto de palavras no texto limpo.

Você vai utilizar os mesmos tokens que criou no último exercício: `lower_tokens`.

Voce precisará fazer download dos seguintes pacotes no console:

`nltk.download('wordnet')` e `nltk.download('omw-1.4')`

Instruções

1. Importe as funções `WordNetLemmatizer` de `nltk.stem` e `Counter` de `collections`.
2. Crie uma lista com somente números alfabéticos de `lower_tokens`. O método `.isalpha()` realizar essa análise.
3. Crie uma outra lista chamada `no_stops` que contem as palavras que não fazem parte do conjunto dentro de `english_stops`.
4. Inicialize um objeto de `WordNetLemmatizer` chamado `wordnet_lemmatizer`.
5. Utilize o método `.lemmatize()` de `wordnet_lemmatizer` nos tokens de `no_stops` e armazene o resultado em `lemmatized`.
6. Cie um bag-of-word utilizando `Counter` chamado `bow` com as palavras lematizadas.
7. print os 10 tokens mais comuns.

Introdução a Gensim

- Lib open source para realizar processamentos relacionados a NLP, assim como NLTK.
- Forma popular de extração quantitativa de atributos de texto para utilização de algoritmos de machine learning.
- Funções prontas para métricas bastante utilizadas, como TF-IDF.

Exemplos

```
from gensim.corpora.dictionary import Dictionary
from nltk.tokenize import word_tokenize
my_documents = ['The movie was about a spaceship and aliens.',
                'I really liked the movie!',
                'Awesome action scenes, but boring characters.',
                'The movie was awful! I hate alien films.',
                'Space is cool! I liked the movie.',
                'More space films, please!',]
```

```
tokenized_docs = [word_tokenize(doc.lower())
                  for doc in my_documents]
dictionary = Dictionary(tokenized_docs)
dictionary.token2id
```

```
{'!': 11,
 ',': 17,
 '.': 7,
 'a': 2,
 'about': 4,
...}
```

Criando um corpus Gensin

```
from gensim.corpora.dictionary import Dictionary
from nltk.tokenize import word_tokenize
my_documents = ['Banana Banana Banana banana Banana a rabit rabit']

tokenized_docs = [word_tokenize(doc.lower())
                  for doc in my_documents]
dictionary = Dictionary(tokenized_docs)
print('Dictionary: ',dictionary.token2id)

corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]
print('corpus: ', corpus)
```

Saída:

```
Dictionary:  {'a': 0, 'banana': 1, 'rabit': 2}
corpus:  [[(0, 1), (1, 5), (2, 2)]]
```

Prática

Questão 3

`code3.py`

Introdução a Gensim

Esta atividade cria um dicionário e corpus gensim! Você usará essas estruturas de dados para investigar tendências de palavras e possíveis tópicos interessantes em seu conjunto de documentos. Para começar, são carregados alguns artigos confusos adicionais da Wikipedia, que foram pré-processados colocando todas as palavras em minúsculas, tokenizadas e removendo stop-words e pontuação. Estes foram armazenados em uma lista de tokens de documentos chamados `articles`. Você precisará fazer um pré-processamento leve e, em seguida, gerar o dicionário gensim e o corpus.

Instruções

1. Importe `Dictionary` de `gensim.corpora.dictionary`.
2. Inicialize o dicionário gensim com os tokens de `articles`.
3. Obtenha o index da string `computer` dentro de `dictionary`. Para fazer isso, utilize o método `.token2id.get()` que retorna os indexes do token de `dictionary`. Utilize `computer` como argumento de entrada.
4. Verifique `computer_id` com a função Print. O método `.get()` de `dictionary` retorna o token relacionado ao id informado no argumento de entrada.
5. utilize list comprehension para iterar em `articles` e criar um MmCorpus de `dictionary`.Crie um bag-of-word com o método `.doc2bow()` de `dictionary` usando `articles` como argumento.
6. Print os 10 primeiros indexes de tokens com sua frequência, no quinto documento.

TF-IDF com Gensim

- Frequência do termo – Frequência inversa do documento
  - Permite determinar as palavras mais importantes em cada documento;
  - Cada corpus pode ter palavras compartilhadas além de palavras irrelevantes;
  - Estas palavras devem ser menos ponderadas em importância;

Fórmula do Tf-idf

```
from gensim.models.tfidfmodel import TfidfModel
tfidf = TfidfModel(corpus)
tfidf[corpus[1]]
```

```
[(0, 0.1746298276735174),
 (1, 0.1746298276735174),
 (9, 0.29853166221463673),
 (10, 0.7716931521027908),
...
]
```

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i})$$

$w_{i,j} =$ Peso do Token i no documento j

$tf_{i,j} =$ Frequência do Token i no documento j

$df_i =$ Quantidade de documentos que contem o token i

$N =$ Quantidade de documentos totais

Exemplo

```
from gensim.corpora.dictionary import Dictionary
from nltk.tokenize import word_tokenize
from gensim.models.tfidfmodel import TfidfModel
my_documents = ['The movie.'
                'Awesome action.'
                'The movie was awful!.'
                'I liked the movie.'
                'More space films, please!']

tokenized_docs = [word_tokenize(doc.lower())
                  for doc in my_documents]
dictionary = Dictionary(tokenized_docs)

corpus = [dictionary.doc2bow(doc) for doc in tokenized_docs]

tfidf = TfidfModel(corpus)

print(tfidf[corpus[0]])
```

Prática

Questão 4

`code4.py`

Gensim e bag-of-words

Agora, você usará seu novo corpus e dicionário gensim para ver os termos mais comuns por documento e em todos os documentos. Você pode usar seu dicionário para pesquisar os termos. Adivinhe quais são os tópicos e sinta-se à vontade para explorar mais documentos!

Você tem acesso ao dicionário e aos objetos de corpus criados no exercício anterior, bem como ao Python defaultdict e itertools para ajudar na criação de estruturas de dados intermediárias para análise.

defaultdict

Nos permite inicializar um dicionário que atribuirá um valor padrão a chaves inexistentes. Ao fornecer o argumento `int`, podemos garantir que qualquer chave inexistente receba automaticamente um valor padrão de 0. Isso o torna ideal para armazenar as contagens de palavras neste exercício.

itertools.chain.from_iterable()

Nos permite iterar através de um conjunto de sequências como se fossem uma sequência contínua. Usando esta função, podemos facilmente iterar através de nosso objeto corpus (que é uma lista de listas).

O quinto documento do corpus é armazenado na variável `doc`, que foi classificada em ordem decrescente e armazenada em `bow_doc`.

Instruções

1. No primeiro loop `for`, print as 5 primeiras palavras com suas respectivas frequências contidas em `bow_doc`, com base em `word_id` e `word_count` que podem ser acessadas em `dictionary`. Lembre da usabilidade do método `.get()` de `dictionary`.
2. Crie um `defaultdict` chamado `total_word_count` em que cada chave são os indexes e os valores são os números de ocorrências nos documentos.
3. Escolha uma chave entre 0 e 5 para exibir a contagem da palavra representada por essa chave.
4. Crie uma lista ordenada de todo o corpus em forma decrescente de `total_word_count`. Utilize método `.itens()` para acessar os elementos. Armazene o resultado em `sorted_word_count`.
5. Exiba os 5 primeiros elementos de `sorted_word_count`.

Prática

Questão 5

`code5.py`

calculando TF-IDF com gensim

Determine os termos significativos para seu corpus aplicando o tf-idf com gensim. Você terá novamente acesso aos mesmos objetos de `corpus` e `dictionary` que utilizou nos exercícios anteriores - `dictionary`, `corpus` e `doc`.

Instruções

1. Inicialize `TfidfModel` chamado `tfidf` usando `corpus` como argumento.
2. Utilize `doc` para calcular os pesos. Voce pode fazer isso passando [doc] para `tiidf`.
3. Print os 5 primeiros elementos de `tfidf_weights`
4. ordene `tfidf_weights` em ordem decrescente.
5. utilizando `dictionary`, exiba os 5 maiores pesos junto com seus respectivos tokens.