Laboratório Nº 4: Text Mining - Bag of Words Model

Extração Automática de Informação
2021/2022

Prof. Joaquim Filipe
Eng. Filipe Mariano

Objetivos

- Conceito de Bag of Words
- Cálculo do TF, IDF e TF-IDF

1. Bag of Words

O conceito de *bag of words* é utilizado no contexto do processamento de linguagem natural e consiste numa forma de extrair características de um texto. A abordagem é muito simples e flexível, que poder ser utilizada de diversas maneiras nessa extração.

Uma *bag of words* é uma representação de texto que descreve a ocorrência de termos num documento. Por termo entenda-se como qualquer n-grama existente no documento, e que no contexto dos laboratórios consiste nos unigramas (1 palavra) e bigramas (2 palavras).

A *bag of words* envolve principalmente dois aspetos:

- Um vocabulário de termos conhecidos
- Uma métrica referente à existência desses termos

É denominado de *bag of words* pelo facto de qualquer informação acerca da ordem ou estrutura das palavras no documento, ser irrelevante. Esta abordagem apenas se preocupa com a ocorrência dos termos no documento, e não em que zonas do texto aparecem. A ideia subjacente é a que documentos semelhantes possuem conteúdo semelhante, logo através desse conteúdo é possível aprender algo acerca do significado de um documento.

A complexidade da *bag of words* é determinada pela análise que se pretende fazer, que pode ser simples ou complexa, dependendo dos termos e dos *scores* de ocorrência dos mesmos.

2. Exemplo de Aplicação de Bag of Words

2.1. Passo 1: Definição do Corpus

De seguida será colocada algumas linhas retiradas de um livro em que se irá assumir que cada linha representa um documento.

```
It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,
```

2.2. Passo 2: Criação do Vocabulário

Retirando a pontuação existente em cada documento, irá ser criada uma lista de todas as palavras (unigrama) existentes, identificando aquelas que são únicas, excluindo as *stopwords*, e aplicando o algoritmo *Porter Stemming*:

- best
- time
- worst
- ag
- wisdom
- foolish

Este foi o vocabulário extraído de um corpus com 4 documentos em que cada documento continha 6 palavras.

2.3. Passo 3: Criar Vetores de Documento

2.3.1. Exemplo 1

O próximo passo consiste em atribuir uma pontuação aos termos em cada documento. O objetivo da criação de um vetor para cada documento é o de tornar cada documento de texto livre num vetor que poderá ser utilizado como input ou output de um sistema de aprendizagem automática.

Uma vez que no passo anterior se determinou que o vocabulário para o corpus era composto por 6 termos diferentes, pode-se assumir esse valor como fixo para o tamanho do vetor de cada documento. O vetor pode ser interpretado como um array de valores em que cada posição corresponde à posição do termo e o seu valor corresponde à pontuação atribuida a esse termo, que pode ser em função da existência ou não do termo (vetor binário), da ocorrência de um termo ou de qualquer outra métrica que se pertenda analisar. O método mais simples para demonstrar a criação de um vetor para cada documento, é a aplicação simples de um método binário para a existência do termo no documento (0 caso não exista e 1 caso exista).

Desta forma, os 4 documentos seriam descritos por vetores contendo 0 ou 1 em cada posição indicando a existência (ou não) desse termo no documento. Para o 1º documento o vetor iria possuir o valor 1 para os termos `best` e `time` e 0 para os restantes:

```
[1, 1, 0, 0, 0, 0]
```

Assim, sendo para os 4 documentos a representação do vetor seria descrita da seguinte forma (as linhas representam os documentos (D_k) e as colunas os termos (T_k)):

	T_1: *best*	**T_2: *time***	**T_3: *worst***	**T_4: *ag***	**T_5: *wisdom***	**T_6: *foolish***
D_1: "It was the best of times,"	1	1	0	0	0	0

	T_1: *best*	T_2: *time*	T_3: *worst*	T_4: *ag*	T_5: *wisdom*	T_6: *foolish*
D_2: "it was the worst of times,"	0	1	1	0	0	0
D_3: "it was the age of wisdom,"	0	0	0	1	1	0
D_4: "it was the age of foolishness,"	0	0	0	1	0	1

Analisando de uma forma mais prática a matriz construída, podemos constatar que o vetor pode ser traduzido através de um array:

```
"It was the best of times," = [1, 1, 0, 0, 0, 0]
"it was the worst of times," = [0, 1, 1, 0, 0, 0]
"it was the age of wisdom," = [0, 0, 0, 1, 1, 0]
"it was the age of foolishness," = [0, 0, 0, 1, 0, 1]
```

Apesar de intuitivo, a métrica de existência ou não de um documento pode ser limitativo, pelo que costumam ser aplicadas outras métricas, tais como: *Term Frequency* (TF), *Inverse Document Frequency* (IDF) e a correlação entre os dois anteriores (TF-IDF).

Utilizando como exemplo a matriz anterior poderemos recorrendo ao TF, construir a mesma matriz mas em que nas células estariam os valores da frequência do termo para um determinado documento.

	T_1: *best*	T_2: *time*	T_3: *worst*	T_4: *ag*	T_5: *wisdom*	T_6: *foolish*
D_1: "It was the best of times,"	0.5	0.5	0	0	0	0
D_2: "it was the worst of times,"	0	0.5	0.5	0	0	0
D_3: "it was the age of wisdom,"	0	0	0	0.5	0.5	0
D_4: "it was the age of foolishness,"	0	0	0	0.5	0	0.5

2.3.2. Exemplo 2

De seguida será mostrado um exemplo em que o conjunto de treino tinha 3 documentos para a classe `Not Happy`. As colunas (unigramas de palavras a azul e bigramas de palavras a verde) representam todos os termos encontrados nos documentos do conjunto de treino da classe *not happy* e as linhas (cor amarela) representam os documentos desse conjunto de treino. Na interseção entre um documento e um termo, temos o valor referente à métrica utilizada para a análise dessa matriz, em que a 1ª e 2ª matriz representam a análise existência dos termos, a 3ª matriz o cálculo do *term frequency* do termo no documento e a 4ª matriz o cálculo do *inverse-document frequency*.

Classe: Not Happy

Binary Vector / 1-Gram

	room	small	messy	breakfast	very	good	few	choices
The room was small and messy.	1	1	1	0	0	0	0	0
The breakfast was not very good.	0	0	0	1	1	1	0	0
Breakfast very few choices.	0	0	0	1	1	0	1	1

Binary Vector / 2-Gram

	room small	small messy	breakfast very	very good	very few	few choices
The room was small and messy.	1	1	0	0	0	0
The breakfast was not very good.	0	0	1	1	0	0
Breakfast very few choices.	0	0	1	0	1	1

TF Vector / 1-Gram

	room	small	messy	breakfast	very	good	few	choices
The room was small and messy.	0,33	0,33	0,33	0	0	0	0	0
The breakfast was not very good.	0	0	0	0,33	0,33	0,33	0	0
Breakfast very few choices.	0	0	0	0,25	0,25	0	0,25	0,25

IDF Vector / 1-Gram

	room	small	messy	breakfast	very	good	few	choices
The room was small and messy.	log (3/1)	log (3/1)	log (3/1)	log (3/2)	log (3/2)	log (3/1)	log (3/1)	log (3/1)
The breakfast was not very good.	log (3/1)	log (3/1)	log (3/1)	log (3/2)	log (3/2)	log (3/1)	log (3/1)	log (3/1)
Breakfast very few choices.	log (3/1)	log (3/1)	log (3/1)	log (3/2)	log (3/2)	log (3/1)	log (3/1)	log (3/1)

3. TF, IDF e TF-IDF

Vamos recordar o cálculo de algumas métricas referidas na última aula, tais como, Term Frequency (TF), Inverse Document Frequency (IDF) e a relação entre ambas:

TF = (Número de vezes que o termo T ocorre num documento) / (Número de termos do documento)

$IDF_{(t)} = \log (N / d_{(t)})$

Em que `t` representa o termo, `N` o número de documentos do corpus e `d` o número de documentos em que o termo `t` ocorre.

$TF\text{-}IDF_{(t)} = TF_{(t)} * IDF_{(t)}$

Em que `t` representa o termo, `TF` a frequência do termo no documento e o `IDF` a frequência inversa do documento para esse termo.

4. Exemplo de Implementação

Criar uma classe `Term` com propriedades `name`, `binary`, `occurrences`, `tf`, `idf`, `tfidf` e `docId` de modo a poder guardar uma estrutura de dados de apoio às *Bag of Words*.

Exemplo:

```
{
    name: "breakfast",
    binary: 0,
    occurrences: 0,
    tf: 0.0,
    idf: 0.001,
    tfidf: 0.0,
    docId: 1
}
```

Tendo uma estrutura semelhante à indicada, é possível mais facilmente manipular os termos, documentos e valores associados a cada métrica, permitindo a realização de cálculos à posteriori a partir dos valores das métricas obtidos, nomeadamente aplicar a soma ou média de de cada uma das métricas em todos os documentos.

Exemplo dos dados referentes ao termo `best` em 3 documentos:

```
[
    {
        name: "best",
        binary: 1,
        occurrences: 4,
        tf: 0.1,
        idf: 0.01,
        tfidf: 0.001,
        docId: 1
    },
    {
        name: "best",
        binary: 0,
        occurrences: 0,
        tf: 0.0,
        idf: 0.01,
        tfidf: 0.0,
        docId: 2
    },
    {
        name: "best",
        binary: 1,
        occurrences: 1,
        tf: 0.05,
        idf: 0.01,
        tfidf: 0.0005,
        docId: 3
    }
]
```

Exemplo de cálculo da soma:

```
{
    name: "best",
    binary: 0, //podia ser considerado apenas 1 se tiver em todos os documentos
    occurrences: 5,
    tf: 0.15,
    idf: 0.01, //não é necessário aplicar soma porque o valor é sempre igual para
o mesmo termo em todos os documentos
    tfidf: 0.0015
}
```

5. Exercícios

1ª Parte

1. Deverá criar um novo módulo `bagOfWords.js` numa diretoria `features`. Nesse ficheiro deverá exportar uma função `addUniqueTerms`, que deverá receber dois arrays de termos (unigramas ou bigramas) e verificar quais os termos que existem no 2ª array que não se encontram no primeiro e adicionar esses termos ao 1º array. A função deverá retornar o 1º array com os novos termos adicionados ao mesmo.

2. No laboratório anterior, elaborou uma função `process` no ficheiro `train.js` que fazia o pré-processamento nos textos (*stopwords*, pontuação, *stemming*, etc), e indicava os unigramas e bigramas de palavras encontrados em cada documento. Agora deverá incorporar a nova função `addUniqueTerms` do módulo `bagOfWords.js` para que a cada documento processado vá adicionando os novos **unigramas de palavras** encontrados. O objetivo é começar com um array vazio e terminar com um array com todos os termos encontrados em todos os documentos, sem repetição. **Esse array deverá ser criado para cada classe**, o que quer dizer que como neste problema existe a classe *happy* e *not happy* deveremos ter no final um vetor de unigramas de palavras para a classe *happy* e outro para a classe *not happy*.

3. Repita o que fez anteriormente para que recorrendo ao módulo `bagOfWords.js` também crie um array de **bigramas de palavras** encontrados em todos os documentos, sem repetição e para cada classe.

4. No módulo `bagOfWords.js` criar uma função `binaryVector` que recebe o array que representa a **bag of words**, e um array de termos e devolve o array de **bag of words** com 1 nos termos encontrados e 0 nos termos não encontrados.

5. No módulo `bagOfWords.js` criar uma função `numberOfOccurrencesVector` que recebe o array que representa a **bag of words** e um array de termos e devolve o array de **bag of words** com o número de ocorrências dos termos encontrados.

6. No módulo `bagOfWords.js` criar uma função `tfVector` que recebe o array que representa a **bag of words** e um array de termos e devolve o array de **bag of words** com o `term frequency` dos termos encontrados.

7. No módulo `bagOfWords.js` criar uma função `idfVector` que recebe o array que representa a **bag of words** e um array de termos e devolve o array de **bag of words** com o `inverse document frequency` dos termos encontrados.

8. No módulo `bagOfWords.js` criar uma função `tfidfVector` que recebe o array que representa a **bag of words** e um array de termos e devolve o array de **bag of words** com o `term frequency - inverse document frequency` dos termos encontrados.

9. No ficheiro `train.js` utilize as funções de criação dos vetores dos exercícios `4`, `5`, `6`, `7` e `8` para os unigramas de palavras encontrados em cada texto nas respetivas classes `happy` ou `not happy`. Como resultado final irão obter para cada documento, os vetores binário, nº de corrências, tf, idf, e tf-idf para unigramas de palavras, ou seja, 5 vetores de unigramas para os documentos da classe `happy` e 5 vetores para os documentos da classe `not happy`.

2ª Parte

1. Crie uma função `sumVector` no módulo `bagOfWords.js` que recebe como parâmetro de entrada um array de objetos do tipo `Term` em que o `name` do termo é o mesmo em todo o array e devolve a soma de todos os elementos das várias propriedades mencionadas anteriormente, excepto do `idf` (que se manterá o mesmo valor) e do `tfidf` (que será a multiplicação da soma dos `tf` pelo `idf`).

2. Crie uma função `avgVector` no módulo `bagOfWords.js` que recebe como parâmetro de entrada um array de objetos do tipo `Term` em que o `name` do termo é o mesmo em todo o array e devolve a média de todos os elementos das várias propriedades mencionadas anteriormente, excepto do `idf` (que se manterá o mesmo valor) e do `tfidf` (que será a multiplicação da soma dos `tf` pelo `idf`).

3. Exporte as funções `sumVector` e `avgVector` do módulo `bagOfWords.js` e utilize-as no módulo `train.js` na função `process` para poder ter um array de objetos do tipo `Term` com os valores das métricas somadas e outro array de objetos do tipo `Term` com os valores das médias das métricas, para cada classe ("happy" ou "not happy").

4. Repita todo o processo realizado no laboratório anterior e os realizados neste laboratório, mas agora para bigramas de palavras.

5. Gravar em base de dados os resultados dos termos e métricas calculadas para o conjunto de treino. Sempre que o processamento do conjunto de treino é feito, estes resultados são recriados na base de dados.

Nota: É esperado que no final desta 2ª parte do laboratório os alunos consigam ter 4 arrays de objetos `Term` em que 2 (somatório e média das métricas) são referentes aos documentos da classe "happy" e os outros 2 (somatório e média das métricas) referentes à classe "not happy", após o pré-processamento de todos os textos de cada classe e aplicação de unigramas e bigramas de palavras únicas para todos os textos pertencentes ao conjunto de treino de cada uma das classes.