O tema que foi atribuído ao nosso grupo e que abordaremos nesta apresentação é o “Spacy’s POS Tagging”.

Começaremos por falar sobre o que é o spaCy, qual a sua organização ou arquitetura e quais as funcionalidades que disponibiliza. De seguida, abordaremos como utilizar a funcionalidade de *POS Tagging* do spaCy, após o que apresentaremos as funções mais importantes do exemplo desenvolvido. Por fim, faremos uma breve apresentação do mesmo.

O spaCy é uma biblioteca *open-source* para o processamento avançado de linguagem natural em *Python*.

As estruturas de dados centrais do *spaCy* são o *Doc* e o *Vocab*.

O objeto *Doc* possui uma sequência de *tokens*, que pode ser uma palavra, pontuação,

espaço em branco, etc., e de *spans*, que são porções do objeto *Doc*, como frases. O *Doc* é construído por um *Tokenizer*, sendo depois modificado pelos componentes de um *Pipeline*. O objeto *Language* coordena estes componentes, ao pegar num texto limpo e enviá-lo para o *pipeline*, retornando um texto anotado, o *Doc*.

O objeto *Vocab* possui um conjunto de tabelas de *look-up*, que tornam a informação comum disponível entre documentos. O *Vocab* é constituído por entradas do tipo *Lexeme*, que são um tipo de palavras sem contexto, ao contrário de uma palavra *token*, que possui vários atributos. Ao centralizar vetores de palavras, atributos léxicos e *strings*, o spaCy evita armazenar múltiplas cópias desses dados, pelo que poupa memória e assegura que existe uma única fonte de verdade.

Também para poupar memória, o *spaCy* codifica todas as *strings* para valores de *hash*, que são acessíveis a partir do *StringStore*.

Apresentaremos agora algumas das funcionalidades mais importantes do spaCy.

* *Tokenization* corresponde à separação de textos em segmentos significativos, como por palavras, pontuação, etc.
* *POS Tagging* é a atribuição de classes gramaticais (como verbo ou nome) a *tokens*.
* *Dependency Parsing* é o processo de obtenção de relações de dependência entre os elementos de uma frase, através de um *parser*.
* *Lemmatization* corresponde à redução de palavras às suas formas base. ‘gostou’, por exemplo, fica ‘gostar’.
* *Sentence Boundary Detection* consiste em separar frases de um *Doc*.
* *Named Entity Recognition* corresponde à atribuição de *labels* pré-definidas a nomes de objetos do mundo real, como pessoas, países, produtos, organizações, etc.
* O *Similarity* compara dois textos e calcula um valor que indica quão similares esses textos são.
* *Text classification* é o responsável pela atribuição de uma *label* a todo o documento ou partes de um documento.
* O *Rule-based matching* procura correspondências entre *tokens* e determinados padrões.
* *Training* corresponde à atualização e melhoria de modelos estatísticos de previsões, utilizando dados de treino.
* O *Serialization* permite guardar objetos do spaCy para ficheiros em disco e carregar objetos previamente guardados.

Como se pode ler nesta frase, dois artigos revistos em 2015 confirmam que o spaCy oferece o *parser* sintático mais rápido do mundo e que a sua precisão está dentro de 1% do melhor disponível. De facto, para além da velocidade e precisão, o spaCy também tem a vantagem de disponibilizar praticamente todas as funcionalidades também facultadas por outas bibliotecas semelhantes.

Podemos apresentar agora a forma básica de obter os *labels* da POS dos *tokens* de um texto. Em primeiro lugar, é necessário instalar o *spaCy*, bem como a linguagem cujo modelo se pretende utilizar. Também se poderia treinar um modelo vazio.

De seguida, no *python*, basta fazer *import* do módulo *spacy* e carregar a linguagem pretendida. Essa linguagem é então utilizada para gerar o *Doc* de um texto. A partir desse *Doc*, consegue-se aceder aos *tokens* e, a partir destes, aos respetivos atributos, incluindo o *part-of-speech*.

Relativamente ao exemplo demonstrativo, desenvolvemos algumas funções auxiliares, que devido à sua relação direta com o spaCy, apresentamos de seguida, com o intuito de aprofundar a forma de utilização dessa biblioteca.

A função generate\_pos\_chart foi criada para gerar um gráfico de ocorrências de cada *tag* POS num determinado *Doc*. Para isso, obtêm-se todas os POS do *Doc* e guardam-se no dicionário tag\_dict, sendo a chave o valor de hash do POS e o valor a *string* percetível pelo ser humano. De seguida, contabilizam-se todas as ocorrências das *tags* POS no *Doc* e divide-se cada uma pelo número de *tokens* total, de forma a obter a respetiva percentagem. A cada um desses valores associa-se a respetiva *tag*. Por fim gera-se uma imagem com um gráfico de barras, ou devolve-se uma lista de listas, em que cada lista interna corresponde ao par *tag* POS – percentagem de ocorrências. Esta lista poderá ser utilizada posteriormente para construir um *Google* *Chart* em *html*.

Na função generate\_information, que obtém informação dos *tokens* de um Doc, simplesmente se percorre esses *tokens* e se obtém o valor de alguns dos seus atributos da mesma forma que foi apresentada no exemplo de utilização básica.

A função generate\_dependencies\_graph é utilizada, como o próprio nome indica, para gerar gráficos de dependências das frases de um *Doc*. De forma geral, simplesmente foi necessário dividir o *Doc* por frases e passa-las a uma função do displaCy, que é um *visualizer*. Essa função é o *server*, quando se pretende obter de imediato o resultado numa página *web*, ou o *render*, quando se pretende obter uma *string* correspondente a conteúdo de uma imagem *svg* ou de um ficheiro *html*. Em ambos os casos, utiliza-se o estilo ‘dep’.

A função generate\_tagged\_text é utilizada para etiquetar entidades. São utilizados dois métodos. Para se obter um resultado mais apelativo, na *web* ou em ficheiros *html*, utiliza-se o mesmo método descrito anteriormente, mas com o estilo ‘ent’. Para se obter o resultado no formato de texto, percorre-se todos os *tokens* do *Doc* e, caso este possua o atributo ent\_type\_, imprime-se o mesmo. Caso contrário, imprime-se só a palavra.

A função add\_tokenizer\_exception permite a adição de novos *tokens* ou modificação de atributos de *tokens* já existentes. Para isso, basta aceder-se ao *tokenizer* do nlp ou linguagem em questão e adicionar-se casos especiais, com o *token* e respetivos atributos.

Relativamente ao exemplo completo, foi utilizado o *flask* como *framework* *web* para providenciar uma interação mais apelativa e intuitiva ao utilizador. Apesar disso, também é possível utilizar as funcionalidades providenciadas na *web* diretamente através da linha de comandos.

Executando então a aplicação, tem-se a página inicial, onde ….

Página Inicial + Exceção Tokens

Páginas de Output

Mafalda

João