Relatório CARD 22 - Prática: Processamento de Linguagem Natural (NLP) (III)

Willian Augusto Soder de Souza

O objetivo deste relatório é apresentar os principais conhecimentos adquiridos ao assistir ao curso 'Natural Language Processing (NLP) Zero to Hero', oferecido pelo canal do TensorFlow no YouTube, e ao curso 'Natural Language Processing with spaCy & Python - Course for Beginners', disponibilizado pelo canal freeCodeCamp.org, também no YouTube. Este relatório será dividido em duas partes: a primeira será referente ao curso oferecido pelo TensorFlow, e a segunda ao curso do freeCodeCamp.org.

Natural Language Processing (NLP) Zero to Hero

Neste curso, o apresentador busca equilibrar a prática e a teoria, explicando cada parte do código de forma teórica. Abaixo segue uma lista dos principais conceitos abordados no curso.

- **Tokenization:** é o processo de dividir um texto em unidades menores chamadas "tokens," que geralmente são palavras. É uma etapa fundamental no Processamento de Linguagem Natural (NLP) porque permite que modelos analisem e processem o texto de forma mais eficiente. A tokenização facilita o entendimento do conteúdo ao separar palavras, frases ou caracteres para análise individual.

- **Sentences to Data:** refere-se ao processo de converter frases ou textos em um formato numérico que os algoritmos de machine learning possam entender e processar.

- **Recurrent Neural Networks (RNNs):** são um tipo de arquitetura de rede neural projetada para processar dados sequenciais, como textos, séries temporais ou qualquer outro tipo de informação onde a ordem dos dados seja importante. No contexto de Processamento de Linguagem Natural (NLP), as RNNs são amplamente utilizadas para tarefas como tradução automática, análise de sentimentos, e geração de texto.

  Como funcionam as RNNs:

  - **Entrada Sequencial:** as RNNs recebem uma sequência de entradas (por exemplo, palavras em uma frase) uma de cada vez. Cada palavra ou token é processado em uma etapa de tempo separada.

  - **Memória e Estados Ocultos:** em cada etapa de tempo, a RNN mantém um estado oculto que captura informações sobre as entradas anteriores. Esse estado oculto é atualizado com base na entrada atual e no estado oculto da etapa anterior. Essa capacidade de manter um estado oculto é o que permite à RNN ter uma "memória" das palavras anteriores na sequência, ajudando a manter o contexto.

  - **Peso Compartilhado:** a mesma função (com os mesmos pesos) é aplicada em cada etapa de tempo. Isso significa que a rede usa o mesmo conjunto de parâmetros para processar toda a sequência, o que é eficiente e facilita o treinamento.

  - **Saída Sequencial:** a RNN gera uma saída para cada entrada na sequência (por exemplo, uma previsão de palavra, sentimento, etc.), ou pode gerar uma única saída após processar toda a sequência (como em classificação de texto).

- **Backpropagation Through Time:** Para treinar a RNN, é usada uma variante do algoritmo de retropropagação chamada Backpropagation Through Time. Neste processo, o erro é propagado para trás através de cada etapa de tempo na sequência, ajustando os pesos para minimizar o erro.

- **Long Short-Term Memory (LSTM):** são uma variante de Redes Neurais Recorrentes (RNNs) que conseguem lembrar de informações importantes ao longo de sequências mais longas. Elas fazem isso através de "portas" internas que controlam o que deve ser lembrado, atualizado ou esquecido, superando problemas como o desvanecimento de gradientes nas RNNs tradicionais. Isso as torna eficazes para tarefas em que a dependência de longo prazo é crucial, como no processamento de linguagem natural.

Natural Language Processing with spaCy & Python - Course for Beginners

Este curso, ao contrário do primeiro, tem um enfoque mais prático e é totalmente voltado para a biblioteca spaCy, utilizada no processamento de linguagem natural. Abaixo, segue um resumo do que é essa biblioteca e dos principais comandos ensinados durante o curso.

- **spaCy:** é uma ferramenta poderosa para processamento de linguagem natural em Python, projetada para análise rápida e eficiente de textos. Ela oferece funcionalidades como tokenização, identificação de partes do discurso, reconhecimento de entidades (pessoas, locais, organizações), análise de dependências sintáticas e representação vetorial de palavras. spaCy é ideal para aplicações que requerem processamento de grandes volumes de texto e compreensão semântica em tempo real.

  Abaixo segue uma lista dos principais comandos desta biblioteca ensinados no curso.

  - **spacy.load:** é usado para carregar um modelo pré-treinado de processamento de linguagem natural. Esse modelo pode ser usado para realizar várias tarefas, como análise gramatical, reconhecimento de entidades e vetorização de palavras.

  - **nlp(text):** transforma o texto em um formato estruturado que facilita a análise e extração de informações.

  - **doc.text:** mostra texto original.

  - **doc.ents:** entidades reconhecidas no texto.

  - **doc.sents:** sentenças segmentadas no texto.

  - **doc[0]:** pega o primeiro token do texto.

  - **token.text:** texto associado ao token.

  - **token.head:** acessa o token ao qual ele está sintaticamente ligado.

  - **token.ent_type:** mostra o tipo de entidade de maneira numérica.

  - **token.lemma_:** mostra a forma base ou raiz da palavra.

  - **token.morph:** mostra a morfologia do token.

  - **doc1.similarity(doc2):** usado para calcular a similaridade semântica entre dois ‘doc’.

- **nlp.add_pipe:** adiciona uma nova pipeline (pipeline é uma sequência de etapas ou componentes que processam e analisam texto de forma organizada e eficiente.).
- **nlp.analyze_pipes:** mostra informações sobre os componentes da pipeline.
- **Matcher:** é uma ferramenta usada para encontrar padrões específicos em textos.
- **spacy.blank:** é usado para criar um modelo spaCy em branco sem pré-treinamento.

Em resumo, esses foram os principais conhecimentos apresentados nos dois cursos. Vale ressaltar que o segundo curso tem um caráter mais prático, e todas as implementações realizadas durante as aulas estão anexadas.

CONCLUSÃO

A compreensão dos conceitos teóricos fundamentais do Processamento de Linguagem Natural (NLP) é crucial para a aplicação eficaz e a inovação neste campo dinâmico. Conhecer as teorias subjacentes, como tokenização, lematização e análise de sentimentos, permite uma abordagem mais informada e estratégica para resolver problemas complexos relacionados ao processamento de texto. Além disso, ter familiaridade com uma biblioteca poderosa como o spaCy amplia significativamente nossas capacidades práticas. O spaCy oferece ferramentas robustas e eficientes para a análise e manipulação de linguagem natural, facilitando a implementação de modelos avançados e a extração de insights valiosos de grandes volumes de texto. Em resumo, a combinação do conhecimento teórico com a habilidade prática em bibliotecas como o spaCy proporciona uma base sólida para o desenvolvimento e a aplicação bem-sucedida de soluções de NLP.