Para rodar o projeto:

1. Acessar a pasta '/relacoes'
2. No terminal executar: '**python3 -m venv .env**' (para setar um ambiente virtual local)
3. No terminal executar: '**source .env/bin/activate**' (nesta pasta o python default sera 3.x)
4. No terminal executar: '**pip install -U pandas spacy**' (instalar dependências)
5. No terminal executar: '**python -m spacy download pt_core_news_sm**'
6. No terminal executar: '**python relacoes.py**' (rodar o código em si)

O dado fornecido para este trabalho estava bem tratada logo não tivemos dificuldades de leitura ou adaptações.

A extração de relações foi implementada utilizando a biblioteca Spacy.

Para cada sentença no dataset foi aplicado o algoritmo de Dependency Parsing.

> *spaCy features a fast and accurate syntactic dependency parser, and has a rich API for navigating the tree. The parser also powers the sentence boundary detection, and lets you iterate over base noun phrases, or "chunks". You can check whether a Doc object has been parsed with the doc.is_parsed attribute, which returns a boolean value. If this attribute is False, the default sentence iterator will raise an exception.*

Deste resultado, navegamos na Parse Tree

> *spaCy uses the terms **head** and **child** to describe the words **connected by a single arc** in the dependency tree. The term **dep** is used for the arc label, which describes the type of syntactic relation that connects the child to the head. As with other attributes, the value of .dep is a hash value. You can get the string value with .dep_.*

Analisamos as dependências da seguinte forma:

Se existe uma palavra relacionada ao argumento 1 e ao argumento 2 ao mesmo tempo, ela é uma potencial relação.

Para identificarmos se realmente é uma relação, vemos qual é a dependência entra elas. Neste caso notamos que as mais comuns eram `nmod` e `appos`.

O arquivo '**resultados.txt**' contém os resultados na ordem das sentenças do dataset fornecido.