**Natural Language 2023/2024**

**Report – 2nd Project**

**Professora Luísa Coheur**

**Grupo 06:** Inês Pissarra nº: 99236 | Madalena Cunha nº: 99267

### 1 - Modelos:

**Jaccard:** Começámos por utilizar um algoritmo de Jaccard simples, com o pré-processamento mais simples (1), que, apesar de já esperarmos uma accuracy baixa, quisemos testar para ganharmos uma maior noção do algoritmo e perceber as suas "falhas".

**TF-IDF & Naïve Bayes:** De seguida, testámos um TF-IDF seguindo de um Naïve Bayes, com pré-processamento simples (1), que apresentou logo melhorias significativas.

**TF-IDF & SVM:** O nosso terceiro modelo foi um TF-IDF seguido de uma Support Vector Machine. Foi neste modelo que começaram a ser testadas outras formas de pré-processamento (1-4). A partir do pré-processamento 4 adicionaram-se bigrams e trigrams ('n_gram=(1,3)'), para que fosse capturada a relação entre as palavras. Adicionou-se também o limite 'max_features=20000' para descartar as mais irrelevantes (evitando overfitting).

**TF-IDF & SVM c/features:** Quisemos experimentar algo diferente e testámos um pré-processamento mais simples (5), em conjunto com duas features: a soma da polaridade das palavras, de maneira a tentar captar melhor se o sentimento geral é negativo ou positivo; a frequência de adjetivos (deceptive reviews costumam apresentar mais adjetivos).

**TF-IDF & SVM c/features c/separação:** Ao modelo anterior, foi adicionada a separação entre Deceptive / Truthful e Negative / Positive. O modelo classifica-as separadamente. Foi o modelo que apresentou melhor accuracy.

**Pré-Processamentos:**

**PP1.** lowercase + stopwords: num momento inicial de pré-processamento, fizemos algo muito simples, colocando todas as palavras em lowercase e retirando todas as stopwords.

**PP2.** lowercase + stopwords (nem todas) + lematização + pontuação: após comparação entre as frases pré-processadas com as técnicas anteriores, constatamos que existiam algumas stopwords que seriam importantes (como not, but, against, ...) e que por isso deixaram de ser retiradas do texto. Ao mesmo tempo decidimos retirar os sinais de pontuação e adicionar uma lematização para juntar as palavras com o mesmo lema.

**PP3.** PP2 + labels para negações: transformámos as palavras no formato '<word>n't' por '<word> not' e depois juntámos as palavras 'not' e 'never' às que lhes sucedem, criando 'NOT_<word>' e 'NEVER_<word>', de maneira a que as palavras passassem a estar relacionadas. Não apresentou melhorias e, portanto, acabámos por abandonar a ideia.

**PP4.** PP2 + stemming + token: nas versões anteriores de pré-processamento era sempre utilizada a função split(), contudo percebemos que a tokenização é mais adequada. O stemming também começou a funcionar melhor nesta versão com tokens.

**PP5.** PP4 exceto lowercase: ao adicionar features ao nosso modelo, o lowercase do pré-processamento tornou-se prejudicial. Para os modelos de features decidimos, portanto, retirá-lo.

Utilizámos sempre a técnica de *Ablation Study* durante o nosso processo, com o objetivo de perceber o impacto de cada pequena alteração ao pré-processamento e às features, e tentar encontrar a melhor conjunção. Neste relatório relatamos apenas as técnicas que se mostraram mais relevantes.

### 2 - Configuração Experimental e Resultados:

Para comparar o modelo *Jaccard* com o *TF-IDF & NB* utilizámos a função train_test_split() com 'test_size=0,2' e 'random_state=0'. O objetivo era analisar a maneira como o segundo apresentava resultados significativamente melhores que o primeiro e, portanto, não fizemos testes tão exaustivos. O modelo *Jaccard* apresentou uma accuracy à volta dos 0,57 e o modelo *TF-IDF & NB* uma accuracy de 0,79.

Os seguintes modelos (*TF-IDF & NB, TF-IDF & SVM,* aplicando a este último os vários pré-processamentos e as features) foram testados utilizando a técnica de 5-fold cross-validation, para tentar englobar tanto os melhores como os piores casos. Cada *test set* representaria assim 20% do ficheiro train.txt, e o *training set* os restantes 80%.

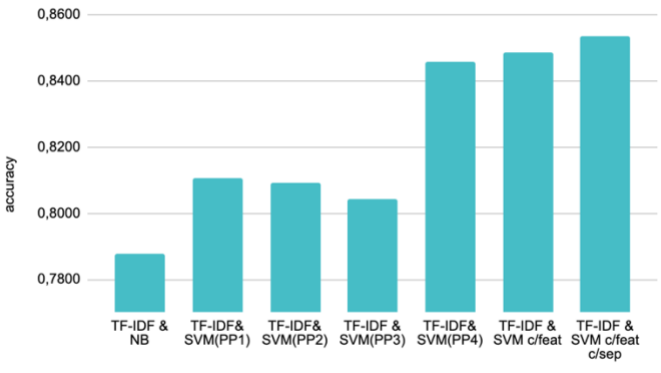
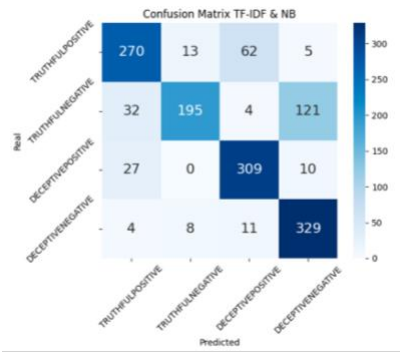
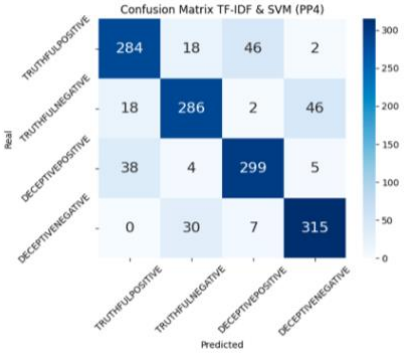
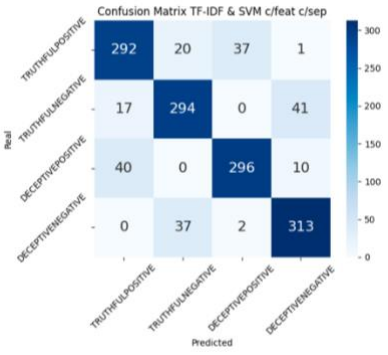



Gráfico 1: Accuracy dos diferentes modelos

Confusion Matrix de alguns dos modelos:




**3 – Discussão:**

Pelos resultados do modelo *Jaccard*, percebemos que facilmente se foca em palavras não tão relevantes. Mesmo retirando stopwords existirão muitas palavras que poderão ter um efeito indesejado. Por exemplo, na review da linha 466: "The Palmer House Hilton hotel takes luxury to a whole new level of existence. (…)" o modelo classificou como NEGATIVE, sendo na verdade POSITIVE. As palavras mais relevantes para esta classificação remetem a algo positivo, mas as restantes são mais em número. Por esta razão ainda testámos tirar nomes (de cidades por exemplo), mas percebemos que essa abordagem poderia ser prejudicial na classificação de outras frases.

E isto leva-nos ao nosso segundo modelo, onde utilizamos a medida TF-IDF que ajudou a que as palavras mais "gerais" tivessem menos peso. O Naïve Bayes, que trabalha com probabilidades, também apresentou ser mais adequado ao problema. Analisando de seguida os dados, pudemos reparar que o modelo continua a apresentar algumas falhas, principalmente na distinção entre TRUTHFUL e DECEPTIVE (verificar confusion matrix da esquerda na secção anterior). Este modelo é sensível a palavras pouco utilizadas e com conjuntos pequenos de dados pode não ter tanta eficácia. No exemplo da linha 7, a verdadeira label seria TRUTHFULNEGATIVE e o nosso modelo classificou como DECEPTIVENEGATIVE. Esta review continha a palavra "uncaring" que apenas aparecia numa outra review (da linha 68) que tinha como label DECEPTIVENEGATIVE. Estas situações acabam por ter impacto no resultado.

Pensámos de seguida que uma SVM poderia se adequar ao projeto, e, após verificarmos uma accuracy melhor, decidimos começar a melhorar o pré-processamento. Remover todas as stop-words "cegamente" poderia estar a tirar alguma informação importante das frases, e então decidimos incluir palavras como 'not', 'but', 'against', o que originou o PP2. No PP2 percebemos que algumas palavras tais como never e not poderiam ser associadas ao negativo (p.e never again), mas existem muitas outras situações em que não o são (review da linha 680 "I've **never** been to Chicago before, but the location was fantastic. (...) I've **never** seen so many options for rooms. (...) We have **never** been treated so much like royalty", foi classificada como NEGATIVE e é POSITIVE). Isto levou-nos ao PP3, no qual foram acrescentadas as labels NOT_word e NEVER_word, o que acabou por não apresentar melhorias. Ao adicionar estas labels, estamos a relacionar o not/never com a palavra seguinte, mas também a tirar força a esta segunda, o que apresentou pioras principalmente na classificação entre DECEPTIVE e TRUTHFUL. Também tem outras fragilidades como o conjunto "if not", que iriam induzir em erro, por exemplo, a linha 905 "if not the best hotel (…)" iria ficar "if NOT_best hotel", o que remeteria a negativo.

Percebemos mais tarde que este problema poderia ser facilmente resolvido adicionando bigrams e trigrams ao nosso TF-IDF. Adicionámos também um limite de features de maneira que o modelo se foque nas mais importantes. Esta mudança originou o modelo TF-IDF + SVM(PP4) que apresentou de imediato melhorias (verificar Gráfico 1).

Por fim, quando foram adicionadas as features, passou-se a captar também o significado das palavras. A separação entre deceptive/truthful e positive/negative funciona melhor neste modelo, pois não nos limitamos à frequência das palavras, e podemos criar novas features a pensar em cada um individualmente. O modelo apresentou ainda falhas principalmente na distinção entre deceptive e truthful, mas pensamos que pode ser melhorado com outras features.

Também reparámos que no conjunto de treino (train.txt) existiam algumas labels que não nos faziam sentido, como por exemplo as das linhas 60 e 116, que se focam em experiências positivas e acabam em "it was a great stay!" e "It was a very enjoyable stay." (respetivamente) e são classificadas como negativas.

Neste relatório falamos dos métodos mais relevantes que utilizámos, mas também experimentámos outros tais como a utilização de um modelo pré-treinado de word2vec. Este não obteve melhores resultados provavelmente por o modelo não estar adaptado ao nosso problema. Para melhorar esta abordagem teríamos de explorar o modelo mais a fundo, adaptá-lo melhor ao problema, ou treinar o nosso próprio word2vec.

**4 - Trabalho Futuro**

Achámos que o projeto foi muito útil para aprofundar o nosso conhecimento e despertar a nossa curiosidade sobre o tema. Por esta razão gostávamos de explorar melhor as features, pois achámos que esse modelo tinha bastante potencial. Adicionalmente, estamos interessados em realizar uma implementação utilizando um modelo pré-treinado como o BERT. Por fim, planeamos tentar implementar de uma rede neuronal, pois estas costumam ser bastante potentes.