Projeto de Ingresso - Processamento de Linguagem Natural 2021

Introdução

Bem-vindo! Se você chegou até aqui, esperamos que esteja considerando fortemente entrar na área de processamento de linguagem natural do Turing USP. Essa área é muito cativante e interdisciplinar. Somos uma das áreas de foco mais diversas do grupo, com um número expressivo de mulheres e pessoas que não fazem ou têm formação superior em cursos de exatas :). Portanto, seja qual for o seu background, não se sinta intimidado. Esse case tem como objetivo te deixar familiarizado com alguns conceitos chave de processamento de linguagem natural. A ideia é te dar uma base para que, depois, a gente consiga desenvolver projetos dentro da área com a contribuição ativa de todos. Caso encontre dificuldades, não hesite em mandar mensagem para os membros mais antigos da área ou mandar mensagens no nosso grupo do whatsapp.

Ao final de cada parte do case, damos referências para que você pesquise sobre os assuntos. Além dessas referências, temos um arquivo extenso de aulas internas dadas dentro da área, que você pode acessar aqui.

O que é Processamento de Linguagem Natural?

NLP (Natural Language Processing) é o estudo relacionado à interação entre computadores e linguagens naturais dos seres humanos. Sendo assim, o foco dessa área de estudo é, de certa forma, ensinar o computador a entender e interpretar as linguagens humanas.

Ao conversarmos com alguém, somos capazes de determinar como a outra parte da conversa está se sentindo, analisando suas palavras e expressões, conseguimos expressar emoções através da maneira como falamos e temos total capacidade de entender o assunto sendo discutido. Tudo isso subconscientemente, não precisamos nos esforçar muito para entender tons negativos e positivos durante uma discussão. Além disso, gírias e abreviações utilizadas tão amplamente em redes

sociais não são desafiadoras para você, e seu cérebro consegue interpretar de forma automática um “vc” como "você" e um “tb” como “também”. Entretanto, essas tarefas, tão simples para nós, são desafiadoras para máquinas.

Como você pode explicar para um computador que a frase “Não estou me sentindo feliz hoje” tem uma conotação negativa, ou que “eu tb ñ gosto de ficar em casa no fds” significa “Eu também não gosto de ficar em casa no fim de semana”? Seria um computador capaz de interpretar frases humanas, com suas singularidades? E seria ele capaz de, em algum nível, até reproduzir a nossa linguagem? Em NLP, estudamos essas relações e queremos que seja possível um computador interpretar, entender e até mesmo reproduzir essas características.

Estrutura do case

Nesse case, nós iremos trabalhar com textos, mais especificamente com um dataset de **reviews de videogames da Amazon**. Esse conjunto de dados advém do dataset Amazon product data. Não se preocupe em baixar os dados da fonte original, utilize **esse link**, alí já separamos o que deve utilizar como dados de treino,validação e teste e fizemos um sample da base de dados. Aqui constam mais informações sobre o dataset original. Esperamos que você já saiba o que são dados de treino, validação e teste!

Em suma, esse dataset contém reviews para jogos vendidos na Amazon. Ele contém o nome do usuário, a nota dada, o timestamp da review, o título dado para a review, o número de upvotes e down-votes e, claro, o texto da review em si. Você terá essencialmente nestes grupos tarefas:

1. Fazer pré processamento do corpus
2. Fazer uma análise extensa do corpus
3. Criar um modelo que dada uma review, prevê sua polaridade (positivo se a nota for maior que três e negativo se a nota for menor ou igual a três).
4. Criação e análise de embeddings com base no corpus

Ferramentas úteis/recomendadas

Recomendamos que a resolução seja feita em Python. **Recomendamos** os seguintes pacotes/ bibliotecas (lembrando que **você pode usar outras libs** além das especificadas aqui):

- Para ler e manipular os arquivos de dados:
  - Pandas
- Para criar visualizações:
  - Seaborn
  - Matplotlib
  - Plotly
- Para pré-processamento e criação de modelos de machine learning:
  - Spacy
  - NLTK
  - Sklearn
  - Gensim
- Se você quiser fazer uma wordcloud ;)
  - Wordcloud

Parte 1 - Pré-processamento

Especialmente quando utilizamos métodos mais antigos de classificação de texto, o comum é aplicar uma série de pré-processamentos ao corpus (conjunto de textos), para que haja uma melhor performance dos modelos.

Isso, entretanto, é uma etapa que não é ortodoxamente feita quando falamos de deep learning. Os passos que são comumente utilizados no pré-processamento são:

1. Tokenização
2. Remoção de stopwords
3. Lematização ou Stemização

O processo de pré-processamento começa com a tokenização. Esse processo consiste em uma forma de separar um pedaço de texto em unidades menores chamadas tokens.

Por exemplo, a frase: Oi, tudo bem?

É quebrada nos seguintes tokens: “Oi”, “,”, “tudo”, “bem”, “?”

Além disso, lembre-se de passar todas as palavras para letra minúscula. Com isso, garantimos que palavras como "oi" e "Oi" sejam interpretadas da mesma forma. Você pode usar as próprias funções do Python para isso.

Também, é importante remover as stopwords são listas de palavras muito recorrentes em línguas, e que normalmente não agregam muito na hora de analisar os textos, e cada língua tem sua própria lista de stopwords, alguns exemplos em inglês são “the”, “is”, “at”.

Uma última explicação seria o que é Stemização ou Lematização. As duas são formas de achar formas básicas de palavras, facilitando que palavras com significado similares sejam identificadas pelo seu algoritmo. Lematização procura uma palavra que existe que possa ser a raiz da palavra que você deseja lematizar, um exemplo seria as conjugações de “ter”, como “tenho”, “tinha”, “tem”, todas seriam reduzidas a “ter”. Stemização por sua vez exige bem menos poder computacional por utilizar um algoritmo mais simples, que apenas corta uma parte da palavra, o que pode resultar em uma forma não dicionarizada desta.

Para fazer isso, você pode utilizar bibliotecas do Python como Spacy e NLTK, ou no R, spacy (ou qualquer outra biblioteca que você queira). Veja as referências no final do documento.

Algumas referências:

- https://medium.com/@gon.esbuyo/get-started-with-nlp-part-ii-overview-of-an-nlp-workflow-7ba1f5948b24
- https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79
- https://www.vooo.pro/insights/tutorial-sobre-expressoes-regulares-para-iniciantes-em-python/
- https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0
- https://towardsdatascience.com/text-preprocessing-with-nltk-9de5de891658
- https://medium.com/voice-tech-podcast/implementing-a-simple-text-preprocessing-pipeline-with-spacy-597a3568bc19

- https://towardsdatascience.com/pre-processing-should-extract-context-specific-features-4d01f6669a7e
- https://medium.com/ensina-ai/introdu%C3%A7%C3%A3o-a-processamento-de-linguagem-natural-174936c096b

Faça o pré-processamento do corpus do dataset com (tokenização), (remoção de stopwords) e (stemização ou lematização)

Parte 2 - Análise do corpus

Essa tarefa não tem um script fixo. A ideia é que você procure explorar o corpus e tirar conclusões com base em análises quantitativas. Você pode verificar a relação entre as notas e algumas características do texto da review. Tais como:

a. Frequência de termos (n-gramas)
b. Tamanho das análises (em número de caracteres
c. Entidades extraídas do texto (por meio de um NER pré treinado por exemplo)
d. Classes gramaticais presentes
e. Proporção de letras maiúsculas

Você também pode incluir as outras features (como o número de upvotes e downvotes), timestamp e etc na sua análise.

Algumas referências:

- https://spacy.io/usage/linguistic-features/
- https://programminghistorian.org/en/lessons/counting-frequencies
- https://m-clark.github.io/text-analysis-with-R
- https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk
- https://monkeylearn.com/text-analysis/
- https://blog.dominodatalab.com/natural-language-in-python-using-spacy/
- https://github.com/turing-usp/Fake-Citation/blob/main/Analysis/EDA.ipynb

Parte 3 - Modelagem e feature engineering

Feature Engineering

Você já deve saber o que é uma feature. As features nos dão informações mensuráveis acerca de um fenômeno observado, elas são uma forma estruturada de guardar informação, de tal forma que essas informações podem ser utilizadas por um modelo.

Essa também é uma etapa que não é ortodoxamente feita quando falamos de deep learning. Por isso, se você for fazer o desafio, novamente, tenha em mente que esse passo não será necessário.

Então, agora que você já tem um texto pré-processado, como transformar isso em features para alimentar um modelo?

Bag-of-Words

Existem várias formas de fazer isso. Para os modelos estatísticos ou de machine learning 'clássico' que vamos sugerir aqui, uma abordagem interessante é a chamada **'Bag of words'**. Com o termo 'bag' (do inglês 'saco' ou 'mochila'), queremos dizer que a ordem das palavras não importa, mas sim o número de ocorrências dessas palavras. Em outros tipos de abordagem, a ordem das palavras pode ser fundamental.

Como a ordem das palavras não é levada em conta, esse tipo de abordagem não permite capturar contexto, num sentido de quais palavras aparecem juntas.

A forma mais convencional de usar o Bag of Words é tabular as frequências com que as palavras acontecem em um dado texto, por exemplo:

Quero comer bolo de cenoura

Quero beber guaraná

Preciso comer feijão. Feijão é bom.

Esses textos poderiam estar, pelos pré-processamentos, da seguinte forma:

`['querer', 'comer', 'bolo', 'cenoura']`

`['querer', 'beber', 'guaraná']`

`['precisar', 'comer', 'feijão', 'feijão', 'ser', 'bom']`

Gerando a seguinte tabulação:

querer	comer	bolo	cenoura	beber	guaraná	precisar	feijão	ser	bom
1	1	1	1	0	0	0	0	0	0
1	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	1	2	1	1

Uma maneira alternativa de criar features é usar um *vetor de componentes binários,* esses vetores codificam, para cada palavra no vocabulário local, a ocorrência ou não desta numa determinada observação. Nós não gravamos a frequência com os quais os termos ocorrem em novos documentos, apenas sua presença ou ausência. Por exemplo:

Meu irmão é muito muito criança, ele lava seus bonecos.

Ele lava o nariz. E o nariz é de criança.

Com os pré-processamentos poderíamos ter os vetores:

`['meu', 'irmão', 'ser', 'muito', 'muito', 'criança', 'ele', 'lavar', 'seus', 'bonecos']`

`['ele', 'lavar', 'nariz', 'nariz','ser', 'criança']`

Gerando as seguintes features para essa observação:

meu	nariz	irmão	ser	muito	criança	ele	lavar	seus	bonecos
1	0	1	1	1	1	1	1	1	1
0	1	0	1	0	1	1	1	0	0

Para alguns modelos, a segunda abordagem pode ser melhor, enquanto para outros, a primeira pode performar melhor.

TF IDF

O valor tf–idf (abreviação do inglês term *frequency–inverse document frequency,* que significa frequência do termo–inverso da frequência nos documentos), é uma medida estatística.

No caso do bag-of-words, a ideia era simplesmente tabular ocorrências. Aqui, não queremos um número de ocorrências puro, mas sim ponderar a importância de cada palavra dado o quanto ela aparece num documento em específico em relação ao corpus (conjunto de documentos) como um todo.

A parte "tf", de *term frequency*, diz respeito à frequência que um certo termo aparece num documento em específico. Essa parte da fórmula dá maior importância a um termo num documento caso ele apareça muito.

$$tf_{x,d} = \frac{nº\ de\ ocorrências\ do\ termo\ x\ no\ documento\ d}{nº\ termos\ no\ documento\ d}$$

A parte "idf", de *inverse document frequency*, diz respeito a uma medida global daquele termo. A ideia é que um termo raro deve receber mais importância do que um termo comum. A palavra "filme" deve aparecer muito num corpus de avaliações do Netflix, mas ela adiciona pouca informação nesse contexto. A fórmula para medir o

"idf" considera a fração dos documentos em que um termo aparece, e faz um cálculo que penaliza isso.

$$idf_{x,D} = \log\left(\frac{nº\ de\ ocorrências\ do\ termo\ x\ no\ conjunto\ de\ documentos\ D}{nº\ documentos\ em\ que\ o\ termo\ x\ aparece\ no\ conjunto\ de\ documentos\ D}\right)$$

Aplique tf-idf ou bag-of-words para criar as features do seu modelo.

Algumas referências:

- https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/
- https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e
- https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Modelos de machine learning

Um passo natural depois de extrair features de um texto é utilizá-las para treinar modelos de machine learning de forma a gerar previsões.

Sugerimos, por exemplo, o uso de modelos do tipo Naïve Bayes. Essa é uma classe de classificadores. Esses modelos são "naive" porque são a mais pura aplicação do famoso teorema de Bayes.

Você pode usar outros modelos do tipo, como boosting, árvores de decisão, regressão logística e afins.

Treine pelo menos dois modelos.

Avalie seus modelos, usando uma ou mais métricas adequadas, que você deve justificar no seu notebook.

Algumas referências:

- https://medium.com/turing-talks/sua-primeira-an%C3%A1lise-de-sentimentos-com-scikit-learn-a47c088ea7bd
- https://medium.com/turing-talks/turing-talks-16-modelo-de-pre
- https://medium.com/turing-talks/turing-talks-17-modelos-de-predi%C3%A7%C3%A3o-decision-tree-610aa484cb05
- https://towardsdatascience.com/decision-tree-algorithm-explained-83beb6e78ef4
- https://medium.com/turing-talks/tagged/modelos-de-predi%C3%A7%C3%A3
- https://pythonprogramming.net/machine-learning-tutorial-python-introduction/

Parte 4 - Embeddings (Word2Vec)

Com o BoW (Bag of Words) our TF-IDF não precisamos de um corpus enorme para conseguirmos bons resultados e estas técnicas também não têm um custo computacional muito alto, porém, o grande problema dessas abordagens é que elas não conseguem capturar contexto. Para isso, usamos embeddings, e nesse caso específico, o Word2Vec!

O que são Word Vectors / Embeddings?

Resumidamente, Word Vectors são vetores que representam o significado de uma palavra. Com isso, conseguimos mapear o contexto em que elas ocorrem e o seu relacionamento com as outras palavras daquele texto, sendo até mesmo possível de se fazer operações matemáticas com elas, como por exemplo:

rei - homem + mulher = rainha

Como funciona o Word2Vec?

O modelo do Word2Vec funciona como uma rede neural de apenas uma hidden layer, sendo que o seu objetivo é aprender os pesos desta camada. Basicamente, a ideia do Word2Vec é a de que se duas palavras têm contextos semelhantes, então elas terão vetores semelhantes também.

É importante saber que existem dois modelos diferentes de Word2Vec: o Skip-Gram e o Continuous Bag of Words. Você pode pesquisar um pouco sobre os dois, mas aqui você deve usar o Skip-Gram.

Treine embeddings com o corpus das avaliações.

Avalie os embeddings com o uso de analogias e similaridades entre palavras. Teste diferentes dimensões.

Algumas referências:

- https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
- https://www.youtube.com/watch?v=ERibwqs9p38 (excelente!)
- https://wiki.pathmind.com/word2vec
- https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/
- https://web.stanford.edu/~jurafsky/slp3/6.pdf

Conclusão

Ufa! Esse projeto foi bastante grande e parabéns por ter chegado até o fim! Esperamos que tenha sido uma experiência enriquecedora e que você tenha gostado!

Agora que você tem algumas das principais ferramentas em NLP, você está apto a fazer projetos e aprofundar seus conhecimentos!