Classificação de Notícias utilizando ML.

Estatidados

Apresentação

- Alex Souza
  - +12 years with Data
    - Today:
      - Data Scientist
      - Teacher
  - IT Governance | Database
  - Master in IA

https://linktr.ee/zouza

Compartilhar...

"A troca de conhecimento é um atalho para o aprendizado." **Alex Souza**

Agenda

Contextualização

- **Introdução**
- **Problema**
- **Objetivo**
- **Solução (Fase 1)**
- **Etapas**

Hands On

- Demonstração
- Luppar News-Rec

Introdução

Diante da grande quantidade de informações geradas no mundo atualmente, aqui iremos focar em notícias (mais especificamente em notícias curtas), podemos encontrar diversas fontes dessas notícias (Reuters, CNN, G1, Diário do Nordeste e etc).

Introdução (Aumento notícias digitais)

Fonte: Poder 360 – dados oficiais do IVC (Instituto Verificação de Comunicação)

Problema

Essas fontes de notícias, costumam classificar as notícias em categorias (aqui iremos utilizar o termo: **rótulos**) e isso pode gerar um GRANDE PROBLEMA, pois nem todas notícias que essas fontes produzem ou recebem, chegam corretamente rotuladas e devido ao grande fluxo dessas notícias, isso fica inviável para um humano rotular manualmente cada uma que chega, sem contar em notícias antigas que possam não ter sido rotuladas devidamente no passado.

O Objetivo

É criar um algoritmo utilizando *Machine Learning* para classificar notícias curtas em rótulos de forma automática, ou seja, o algoritmo recebe uma notícia e informa de qual rótulo (categoria) é aquela notícia.

O Objetivo (Exemplo)

Exemplo de Notícia: *“Pesquisa Datafolha divulgada pelo jornal “Folha de S.Paulo” nesta quarta-feira (8) aponta que 69% dos entrevistados dizem que vão perder renda durante a crise sanitária provocada pela epidemia de coronavírus no Brasil. O levantamento mostra ainda que 76% defendem que as pessoas fiquem em casa para evitar que o vírus se espalhe.” Abril-2020*

Rótulos: Saúde, Finança.

A Solução (1° Fase)

O foco foi em classificar as notícias em apenas um rótulo (**monorrótulo**), a segunda fase (multirrótulo) já esta em desenvolvimento e em breve estarei aqui apresentando!

1° Fase – O que foi utilizado...

Web Scraping (Beautiful Soup - Python) pesquisando no site: G1 Notícias e aplicando a “raspagem”, em seguida realizando uma **limpeza e tratamento** desses dados. Como resultado, foi criada a fonte de dados: **z6News** contendo 34.327 notícias divididas em 6 categorias (esporteNews, politicaNews, tecnologiaNews, financaPessoal, educacaonews, cienciianaturezasaudenews)

Download da Fonte

1° Fase – O que foi utilizado...

Uso **Processamento de Linguagem Natural** ou do inglês *NLP — Natural Language Processing* é um campo de Inteligência Artificial que dá às máquinas a capacidade de ler, entender e extrair significado das linguagens humanas.

Fonte

1° Fase – O que foi utilizado...

Algoritmos de Classificação - Supervisionados

- ***KNN (K-Nearest Neighbor)***
  - k = 5;
  - Distância Euclidiana.
- ***SVM (Support Vector Machines)***
  - *Kernel* = RBF (Função base Radial);
  - C = 1.0;
  - *gamma* (γ) = 0.01.
- ***DT (Decision Tree)***
  - *min_samples_split* = 40. (divisão de um nó interno)
- ***RF (Random Forest)***
  - *min_samples_split* = 40;
  - *n_estimators* = 10 (árvores da floresta).

SVM | KNN | Decision Tree | Random Forest

Biblioteca Scikit-Learn do Python

1° Fase – O que foi utilizado...

Representações de Documentos (*Tradicionais)*

BAG-OF-WORDS (BOW) E TF-IDF [1] [2]

1. *I love dogs.*
2. *I hate dogs and knitting.*
3. *Knitting is my hobby and my passion.*

	i	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

$$\text{tf-idf}_{t,d} = (1 + log\ \text{tf}_{t,d}).log\frac{N}{\text{df}_t}$$

	i	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	0.18	0.48	0.18							
Doc 2	0.18		0.18	0.48	0.18	0.18				
Doc 3					0.18	0.18	0.48	0.95	0.48	0.48

[1] ***TF**: Frequência do Termo no Documento - **IDF**: Inverso da Frequência do Termo nos Documentos*

[2] ***Problemas**: alta dimensionalidade e esparsidade, ambiguidade semântica.*

Biblioteca Scikit-Learn do Python

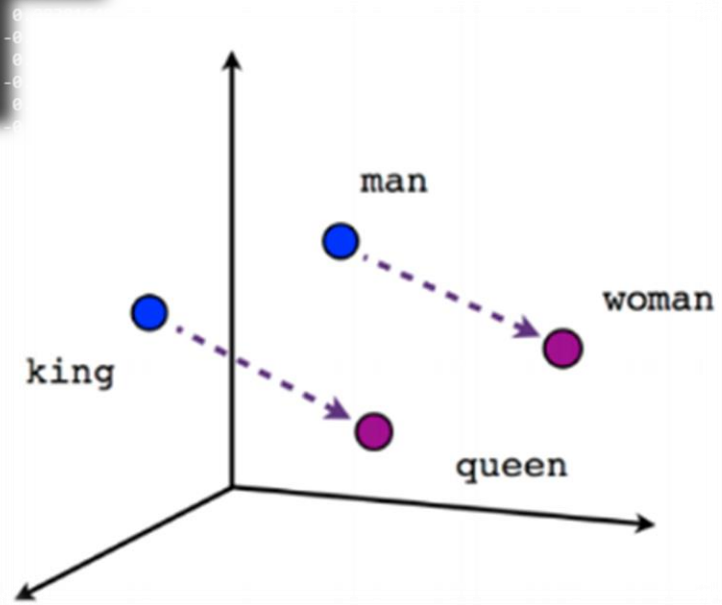
1° Fase – O que foi utilizado...

Representações de Documentos (*Word2Vec e FastText)*

Representa um termo por um vetor de números reais, denso e de tamanho arbitrário.
Sentido semântico dos termos (próximos no vetor);

```
# Consultando o vetor embedding de uma das palavras
w2v['internaco']
```

```
array([ 0.0827674 ,  0.08832473, -0.01311889,  0.02288111,  0.08373141
        0.02018465, -0.00747525, -0.2001954 , -0.00445932, -0.02290371
       -0.10552743,  0.05140657, -0.04853147, -0.11712656, -0.01261191
       -0.05801427,  0.0759929 , -0.05284167,  0.04576398, -0.00043701
        0.05200208,  0.05424974,  0.07770283,  0.14550638,  0.01520923
        0.08429807,  0.07875729, -0.21486288,  0.11415743, -0.20992391
       -0.0685881 ,  0.03464196,  0.06639262,  0.0711642 , -0.02454626
        0.08453867, -0.19376495, -0.11627585, -0.09920968, -0.08798337
       -0.04663217, -0.00564719, -0.07723233,  0.05682064, -0.0043568
        0.04893576, -0.08547731, -0.0648665 ,  0.04425338, -0.03170114
        0.06667162, -0.02683596,  0.01788042,  0.14344227, -0.03635204
        0.14050098,  0.04153308, -0.003143  , -0.04599666,  0.10887373
        0.00666679,  0.1045256 , -0.1121958 ,  0.23045965,  0.04365543
       -0.02381746,  0.05996049,  0.17308174, -0.03465504,  0.03182534
       -0.05723094,  0.04241265, -0.02814974, -0.0014271 ,  0.16333932
       -0.0364379 , -0.0770235 , -0.11705896, -0.0410427 ,  0.06001365
       -0.04882436,  0.03046485, -0.05298632, -0.0689922 ,  0.05081243
        0.05403685, -0.06309855, -0.15442127, -0.02937055,  0.08284083
       -0.06815892,  0.08247626,  0.0269919 , -0.07661436,  0.03036283
        0.06985363, -0.01942973, -0.00918825,  0.00975103,  0.09405516
       -0.14607732,  0.06632613, -0.14079218,  0.11086603, -0.06085347
        0.10566825,  0.03228668,  0.03706625, -0.04345088,
       -0.0652489 ,  0.03477562, -0.03389035,  0.01364257,
        0.01293027, -0.06507182,  0.02537419,  0.03500558,
       -0.09634838, -0.09146278,  0.10349708,  0.01920207,
       -0.11610682, -0.01185585,  0.05889346,  0.00683097,
        0.05101144,  0.0070029 , -0.05513643,  0.01780204,
```



1° Fase – O que foi utilizado...

Representações de Documentos (*Word2Vec e FastText)*

A partir do vetor *embedding* dos termos é necessário obter uma **representação de documentos**. Normalmente um documento é representado pelo vetor média dos vetores dos termos que o compõe (*Word2Vec*).[3]

	-	-	-	-	-	-	-	-	...	-
Doc 1	-0.82	-0.04	0.67	0.35	-0.30	-0.66	1.25	-1.11	...	1.63
Doc 2	1.32	0.98	-0.10	-1.48	0.23	-0.71	-0.87	-0.18	...	-0.89
Doc 3	-0.51	1.54	-0.56	0.67	-0.32	-0.42	1.99	0.91	...	-1.21

[3] *KIM, H. K.; KIM, H.; CHO, S. Bag-of-concepts: Comprehending doc. rep. through clustering words in distr. repres.Neurocomputing, Elsevier, 2017.*

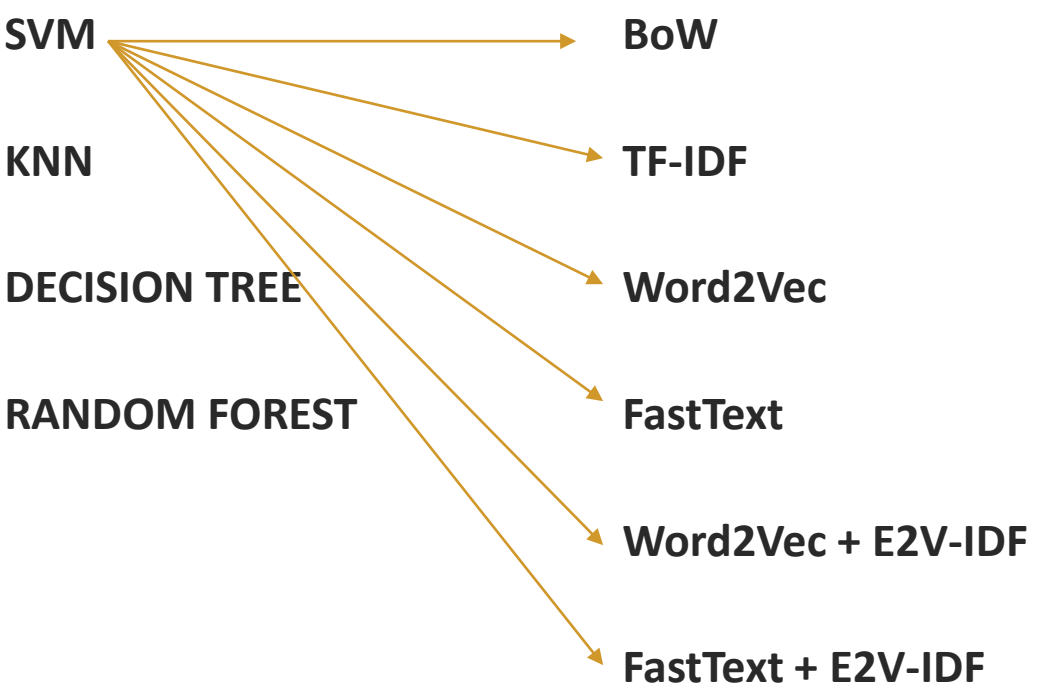
Biblioteca GENSIM

1° Fase – Os testes

Base de dados com 34.327 notícias curtas

Os algoritmos de classificação foram mesclados com as Representações de Documentos.

As combinações mais bem avaliadas foram: ***SVM(RBF)+W2V-IDF*** *e* ***SVM(RBF)+BoW****.*



1° Fase – Os resultados

- **model (f1-score)**
- **SVM(RBF)+BoW (0.806741)**
- **SVM(RBF)+W2V-IDF (0.781892)**
- SVM(RBF)+FT-IDF (0.774696)
- SVM(RBF)+TFIDF (0.773152)
- RF+BoW (0.768957)
- RF+TFIDF (0.759868)
- KNN+TFIDF (0.759518)
- KNN+W2V-IDF (0.752294)
- KNN+W2V (0.746992)
- KNN+FT-IDF (0.742418)
- SVM(RBF)+W2V (0.740525)
- KNN+FT (0.740292)
- SVM(RBF)+FT (0.738165)
- RF+W2V-IDF (0.732630)
- RF+W2V (0.730999)
- RF+TF-IDF (0.721182)
- RF+TF (0.719608)
- DT+BoW (0.679319)
- DT+TFIDF (0.657645)
- KNN+BoW (0.652606)
- DT+W2V-IDF (0.640516)
- DT+W2V (0.636350)
- DT+FT-IDF (0.624523)
- DT+FT (0.620765)

Utilizando a métrica *f1-score* com *Cross-Validation* de 10 *Folds*

1° Fase – Visualizando a performance por rótulos

Aplicando a combinação **SVM(RBF)+W2V-IDF** (ou seja, o algoritmo de classificação: **SVM** em conjunto com a representação de documentos: **Word2Vec** utilizando a Abordagem **E2V_IDF**), os resultados do modelo foram os seguintes quando utilizamos 80% do *dataset* *(**z6News**)* para Treinamento e 20% para teste:

```
Precision: 0.8612184796613289
Recall...: 0.6814739295077192
F1-Score.: 0.7608748678754371
Accuracy.: 0.6746286047189047
              precision    recall  f1-score   support

           0       0.90      0.96      0.93      1008
           1       0.80      0.83      0.82      1319
           2       0.58      0.78      0.66       735
           3       0.33      0.85      0.48       408
           4       0.72      0.89      0.79       925
           5       0.69      0.85      0.76      1038

   micro avg       0.68      0.86      0.76      5433
   macro avg       0.67      0.86      0.74      5433
weighted avg       0.72      0.86      0.78      5433
 samples avg       0.68      0.68      0.68      5433
```

Notícias que tem um "pé" em finanças por exemplo e outro em política

***Rótulos**: esporteNews(0), politicaNews(1), tecnologiaNews(2), financaPessoal(3), educacaonews(4), cienciaNaturezaSaudenews(5)*

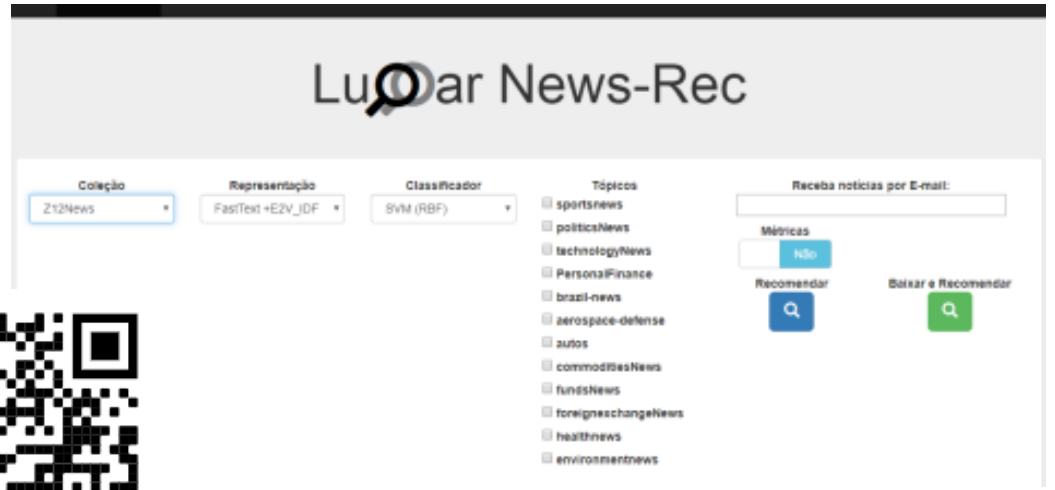
1° Fase – Código Fonte

Código Fonte *(versão didática de demonstração do funcionamento interno)*

O código fonte esta disponibilizado no **GitHub**, assim como a fonte de dados criada (**z6News**).

Trabalho completo (confiram!!!)

- Documentação
- Aplicação
- Fontes



Agenda

Contextualização

- Introdução
- Problema
- Objetivo
- Solução (Fase 1)
- Etapas

Hands On

- **Demonstração**
- Luppar News-Rec

Demonstração

```
def benchmark_new_f1(model, X, y):
    scores = []
    kf = KFold(n_splits=10, random_state=66, shuffle=True)
    kf.get_n_splits(X, y)
    for train, test in kf.split(X, y):
        X_train, X_test = X[train], X[test]
        y_train, y_test = y[train], y[test]
        scores.append(f1_score(model.fit(X_train, y_train).predict(X_test)
        print (pd.DataFrame(scores)) # Guardar dados das 10 rodadas
    return np.mean(scores)
```

```
# Criando a função para a métrica Acurácia
from sklearn.model_selection import KFold
def benchmark_new(model, X, y):
    scores = []
    kf = KFold(n_splits=10, random_state=66, shuffle=True)
    kf.get_n_splits(X, y)
    for train, test in kf.split(X, y):
        X_train, X_test = X[train], X[test]
        y_train, y_test = y[train], y[test]
        scores.append(accuracy_score(model.fit(X_train, y_train).predi
        print (pd.DataFrame(scores)) # Guardar dados das 10 rodadas
    return np.mean(scores)
```

Agenda

Contextualização

- Introdução
- Problema
- Objetivo
- Solução (Fase 1)
- Etapas

Hands On

- Demonstração
- **Luppar News-Rec**

- Pode escolher a **Coleção**, **Representação** e o **Classificador**
- Escolhe **Tópicos** que deseja receber a recomendação
- **Métricas** – Se Sim, mostra as métricas na tela
- **Baixar e Recomendar** – Vai no site de notícias, no caso em questão (Reuters), baixa notícias das últimas 24hs e essas notícias são classificadas com base no modelo criado a partir da combinação: Representação de Documento e Classificador para a essa coleção e mostra na tela as notícias que o modelo classificou com o(s) tópico(s) escolhido(s)
- **Recomendar** – Recomenda com as últimas notícias que tiver no servidor

Referências

- LUPPAR NEWS-REC: UM RECOMENDADOR INTELIGENTE DE NOTÍCIAS
- LUPPAR: UM SISTEMA DE RECUPERAÇÃO DE INFORMAÇÃO PARA COLEÇÕES FECHADAS DE DOCUMENTOS
- Agente Inteligente para Classificação de Notícias por Assunto (Artigo)
- Blog do Alex Souza
- Luppar News-Rec (Site do Recomendador)

https://linktr.ee/zouza

Obrigado

Estatidados