Treinamento LabPi

Seja bem-vindo/bem-vinda! Esse é um cronograma de estudos introdutório feito para os novos integrantes do LabPI. Com esse cronograma, você aprenderá a base de conhecimento necessária para as pesquisas que são realizadas em nosso laboratório. Bons estudos!

IMPORTANTE: Os artigos utilizados neste treinamento, tanto os produzidos pelo laboratório quanto os de outros autores, estão disponíveis no Drive Papers Treinamento. Caso algum artigo não esteja lá ou você não consiga acessá-lo, entre em contato com alguém do laboratório para solicitar o acesso.

Início

Primeiramente, tenha em mente que Ciência de Dados, Inteligência Artificial, Machine Learning e Deep Learning fazem parte do mesmo universo, mas definem assuntos diferentes e que um está contido no outro. Para entender melhor os conceitos separamos alguns vídeos que explicam melhor essa diferença:

- Qual a diferença entre Inteligência Artificial, Machine Learning, Data Science, Dee…
- O que é MACHINE LEARNING? Introdução ao APRENDIZADO DE MÁQUINA | M…
- Machine Learning Explicado
- Machine Learning (O que faz o ChatGPT funcionar) // Dicionário do Programador

Tecnologias utilizadas

Em nossas atividades utilizamos majoritariamente a linguagem Python por ela já oferecer várias bibliotecas com soluções já implementadas de ML, IA, representação, entre outras. Isso nos poupa tempo durante a pesquisa. No início pode parecer difícil, mas logo você verá que Python não é tão complicado assim e que, aos poucos, o conhecimento vai fazendo mais sentido e você vai entendendo os problemas mais complicados.

Como você já sabe programar (pelo menos em C, já que está no laboratório), um primeiro passo para aprender Python é entender a sintaxe da linguagem e aprender a como executar os códigos em Jupyter. Esses vídeos podem ajudar:

- MELHOR FORMA DE APRENDER PYTHON (Google Colab Notebook)
- Curso Completo de Python - do Zero ao Avançado (Masterclass)

É recomendado o uso do Git para ter o versionamento do código (pois nunca se sabe quando a luz/internet do CTAN irá cair ou um servidor parar de funcionar):

- GIT: Mini Curso para Você Sair do Zero! (Aprenda em 45 Minutos)

Na maioria dos projetos executamos os códigos nos servidores do laboratório, o que requer outros conhecimentos como a criação de ambientes virtuais, instalação de drivers específicos, versionamento de bibliotecas, entre outros temas que você aprenderá posteriormente. Algumas leituras que podem te ajudar:

- Ambientes virtuais em Python
- Pyenv: Guia completo de Instalação e Configuração
- Como gerenciar diversas versões do Python e ambientes virtuais

Um material complementar interessante e uma abordagem mais "mão na massa" é o livro Hands-on Machine Learning with Scikit-Learn: (PS.: caps de 1-5 são muito importantes)

- 2-Aurélien-Géron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tens…

NLP

O Processamento de Linguagem Natural (NLP) é um campo interdisciplinar que combina ciência da computação, inteligência artificial e linguística computacional para estudar a interação entre computadores e a linguagem humana. O objetivo principal do NLP é permitir que as máquinas compreendam, interpretem e respondam à linguagem humana de maneira útil e significativa. O laboratório tem diversos trabalhos na área de NLP:

- (PDF) CluWords: Exploiting Semantic Word Clusters for Enhanced Topic Modeling
- Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling - ScienceDirect
- An Effective, Efficient, and Scalable Confidence-Based Instance Selection Framework for Transformer-Based Text Classification
- Evaluating Topic Modeling Pre-processing Pipelines for Portuguese Texts | Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)

Exercício 1: Detecção de Spams

Esse exercício tem como objetivo te ensinar as técnicas básicas de NLP e mineração de dados. Aqui você verá um pouco da tarefa de classificação de texto, indo desde a representação dos dados até uma avaliação de classificadores tradicionais.

Passo 1: Análise dos dados

Inicialmente, para você ir se familiarizando com o Python, iremos fazer apenas uma análise inicial dos dados. Utilizaremos uma base de dados de emails que são categorizados em spam e ham. Ao baixar e descompactar a base, você encontrará dois arquivos: um readme com informações úteis e um arquivo sem extensão "SMSSpamCollection" que é a base de dados no formato tsv (separada por tab).

Carregue a base de dados no Python (use pandas ou outra biblioteca parecida) e gere uma análise inicial da base de dados: plotar a distribuição de classes, número de palavras por documentos, número de documentos, etc.

Ex. de análise de dados:

- https://developers.google.com/machine-learning/guides/text-classification/step-2?hl=pt-br

A base de dados encontra-se no link:

- https://archive.ics.uci.edu/ml/machine-learning-databases/00228/smsspamcollection.zip

Passo 2: Pré-processamento

Uma parte muito importante da tarefa de classificação de texto e da NLP, no geral, é o pré-processamento dos dados, o artigo Evaluating Topic Modeling Pre-processing Pipelines for Portuguese Texts | Anais do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia) explica um pouco mais sobre isso. Para esse passo, faça o pré-processamento da base de dados.

Passo 3: Representação TF-IDF

Agora que você já tem os dados pré-processados chegou a hora de vermos sobre como representar os textos de uma maneira que os modelos de classificação irão entender. Existem diversas maneiras de representar um texto, indo desde técnicas mais simples, como TF-IDF, até técnicas mais avançadas, como embedding estáticos e embedding contextuais. Estudos recentes mostram a importância de se ter uma boa representação dos dados, como apresentado em On the class separability of contextual embeddings representations – or "The classifier does not matter when the (text) representation is so good!", quando utilizamos uma representação boa o suficiente o classificador não importa.

Nesse passo crie uma representação TF-IDF dos seus dados pré-processados. Links úteis:

- From Traditional to Modern: A Comprehensive Guide to Text Representation Techniques in NLP | by Susovan Dey
- Wikipedia TF-IDF
- https://developers.google.com/machine-learning/guides/text-classification/step-3?hl=pt-br

Passo 4: Classificação

Agora que já temos as representações dos nossos textos podemos classificar, existem diversos modelos de classificação que podem ser utilizados. Aqui iremos começar com três deles: KNN, SVN e árvore de decisão. Implemente esses modelos e teste em nossa base de dados. Para avaliar os modelos use as mais comuns, acurácia e Macro F1. Links úteis:

- 1.10. Decision Trees — scikit-learn 1.5.1 documentation
- Como o KNN funciona
- 1.4. Support Vector Machines — scikit-learn 1.5.1 documentation
- accuracy_score — scikit-learn 1.5.1 documentation
- f1_score — scikit-learn 1.5.1 documentation (Pegue a Macro F1)

Passo 5: Ajuste de hiperparâmetros

Como você pode ter percebido, todos esses métodos de classificação tem diversos parâmetros que podem ser ajustados e que podem fazer diferença na eficácia do modelo. Para acharmos o melhor conjunto de parâmetros, podemos utilizar a técnica de Grid Search (existem outras). Para cada método, pesquise quais parâmetros podem fazer mais diferença na eficácia do modelo se variados e use o Grid Search para encontrar o melhor conjunto de parâmetros. Link útil:

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV

Passo 6: Qual o melhor classificador?

Agora que você implementou, testou e avaliou 3 classificadores, já podemos escolher o melhor? Não, ao executar apenas uma vez cada classificador não podemos afirmar que um método é melhor que o outro, pois há muitos fatores que podem influenciar nessa escolha: variáveis randômicas dos métodos, quantidade de dados usados para o treinamento, entre outros.

Para conseguirmos fazer uma comparação mais justa, devemos executar os métodos mais de uma vez, mas não simplesmente executar novamente com o mesmo treino e teste. Temos técnicas específicas para fazer essa validação dos dados, as técnicas que mais utilizamos são K Fold cross validation e Holdout. Links úteis:

- A Gentle Introduction to k-fold Cross-Validation - MachineLearningMastery.com
- StratifiedKFold — scikit-learn 1.5.1 documentation
- KFold — scikit-learn 1.5.1 documentation

Depois de utilizarmos o Kfold (Holdout), teremos N execução de cada método. Para mostrarmos esses resultados podemos utilizar a média das N execuções juntamente com o intervalo de confiança (desvio padrão).

Por fim, para conseguirmos comparar se um método é melhor/pior que outro necessitamos de realizar algum teste estatístico. Normalmente utilizamos o Teste T com correção de Bonferroni como teste estatístico e ele nos retorna para cada par de métodos se eles são estatisticamente equivalentes ou não.

Execute novamente todos os modelos agora fazendo Validação cruzada (com 10 folds) e teste estatístico.

Passo 7: Outras formas de representação

Aqui você já conhece a base da classificação de texto, já aprendeu sobre os classificadores, uma forma de representação simples, como ajustar os hiperparâmetros e como fazer uma comparação entre métodos de forma correta. Agora iremos estudar técnicas mais refinadas para melhorar a classificação. Primeiramente teste outras técnicas de representação textual. Use as representações word2vec e fast Tex e compare os resultados com a representação tf-idf

- Word Embeddings for NLP
- Intuitive Guide to Understanding GloVe Embeddings | by Thushan Ganegedara | Towards Data Science
- Using FastText models for robust embeddings

Passo 8: Modelos transformers

Atualmente, na área de classificação de texto, o estado da arte são modelos baseados em transformers. Para começar a compreendê-los melhor, leia os materiais abaixo e faça a classificação zero-shot do BERT. Agora, pegue a representação gerada pelo BERT zero-shot e use nos classificadores usados anteriormente.

- The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) – Jay Alammar
- Deconstructing BERT: Distilling 6 Patterns from 100 Million Parameters
- Deconstructing BERT, Part 2: Visualizing the Inner Workings of Attention

- Text Classification with BERT
- 6 Steps to Build RoBERTa (a Robustly Optimised BERT Pretraining Approach)
- Classifique o texto com BERT
- BERT
- On the Cost-Effectiveness of Neural and Non-Neural Approaches and Representations for Text Classification: A Comprehensive Comparative Study

Passo 9: Fine-tuning

Faça o fine-tuning do BERT em sua base de dados. Para isso, utilize instâncias com GPU do google colab ou rode nos servidores do laboratório (sua máquina não aguenta executar isso). Com o modelo fine-tuning, use-o para classificar os dados (faça o fine-tuning **apenas** com a partição de **treino**). Agora, use a representação junto com os outros classificadores e compare os resultados.

- Fine-tuning BERT for Text Classification: A Step-by-Step Guide | by Technocrat | Medium
- Fine-tuning BERT for Text classification | Kaggle
- Fine-tuning BERT for Text Classification
- Fine-Tuning BERT using Hugging Face Transformers

Passo 10: Outras Bases

Se chegou até aqui, você implementou 5 classificadores (KNN, SVN, árvore de decisão, BERT zero-shot e BERT fine-tune) e 5 formas de representar um texto (TF-IDF, word2vec, fast Taxt, BERT zero-shot e BERT fine-tune), mas rodou todos os experimentos em apenas uma base de dados. Peça para algum outro integrante do LabPi para te passar mais 2 bases de dados já formatadas no padrão utilizado pelo laboratório e execute o que foi pedido até aqui nelas. Como são muitos experimentos, esse é um bom momento para largar os jupyter notebook e começar a organizar seu código em .py. Crie uma estrutura de código modular e que seja possível de executar todos os experimentos apenas variando os argumentos de entrada dos arquivos, aprenda a utilizar arquivos bash e screens para conseguir deixar os códigos executando nos servidores mesmo com sua máquina não conectada a eles.

Passo 11: Avaliação

Tabele todos os resultados que teve até aqui e veja se conseguiu chegar às mesmas conclusões dos artigos que leu anteriormente. Qual dos modelos obteve o melhor resultado? Qual combinação de representação+classificador foi a melhor?

Passo 12: Extras

Atividades extras que são interessantes, mas não essenciais (dependendo de qual trabalho você fará parte, serão essenciais sim).

- Plotar a representação visual com TSNE, PCA ou UMAP das representações textuais para ver qual delas é possível obter uma melhor separabilidade das classes.
- Utilizar Seleção de instâncias para reduzir o número de elementos de treinamento dos modelos sem que os mesmos percam a eficiência.
- Utilizar undersampling para balancear o número de documentos em todas as classes. Avaliar os modelos em relação ao seu viés para classe majoritária.

- Leia mais sobre modelos baseados em transformers mais atuais (LLMs) como LLama, GPT, Bloom, como utilizá-los, como fazer o fine-tuning, …
- Pesquise sobre as outras áreas de NLP além de classificação, como análise de sentimentos (Feita de maneira não supervisionada), few-shot de modelos transformers
- Faça os exercícios de Modelagem de Tópicos
- Faça os exercícios de recomendação

Exercício 2: Modelagem de Tópicos

Neste exercício, você aprenderá a utilizar umas das principais técnicas de NLP para mineração de dados em um conjunto de documentos. A modelagem de tópicos consiste em, dado um conjunto de documentos, extrair os principais temas - ou como o próprio nome diz, tópicos - presentes nos documentos.

Passo 1: Entendendo o funcionamento dos modelos

Inicialmente, você deve entender como os principais modelos funcionam e quais técnicas utilizam para realizar a modelagem de tópicos. Para isso, você pode ler um pouco mais sobre eles:

- https://cognitivemachine.medium.com/topic-modelling-techniques-f1ce0d0c3262
- https://medium.com/@n83072/topic-modeling-bertopic-ca1b73a035f2
- https://www.freecodecamp.org/news/advanced-topic-modeling-how-to-use-svd-nmf-in-python/
- CluHTM - Semantic Hierarchical Topic Modeling based on CluWords

Perceba que CluHTM é uma técnica criada pelo Felipe Viegas, ex-aluno do labPI e LBD!

Passo 2: Pré-processamento

Como você aprendeu no exercício anterior, o pré-processamento dos documentos é uma fase muito importante de qualquer experimento. Então, utilizando a base TREC aplique as etapas de pré-processamento que você já utilizou. Analise o tamanho e a densidade dos documentos antes e depois desse processo.

Passo 3: NMF

O primeiro modelo a ser utilizado na modelagem de tópicos é o NMF (Non-Negative Matrix Factorization). Entenda como utilizar o modelo aqui e implemente sua versão. Em seguida, realize as seguintes análises:

- Faça a modelagem de tópicos com variação na quantidade de tópicos (5 e 10);
- Varie a quantidade de palavras em cada tópico (10 e 20);
- O que você percebeu sobre variar tanto a quantidade de tópicos quanto de palavras? O que isso muda nas matrizes do modelo? Quais resultados são mais interpretáveis qualitativamente?

Passo 4: BERTopic

Agora, você deve utilizar o BERTopic, um modelo que aplica técnicas *bem* diferentes do modelo anterior, para realizar a modelagem de tópicos, realizando as mesmas variações propostas no exercício anterior. Para entender como utilizar o modelo, leia aqui e, caso sinta necessidade de mais explicações, a documentação do BERTopic é bem completa! Além disso, realize as seguintes análises:

- Além de um arquivo com as palavras presentes em cada tópico, utilize técnicas de visualização próprias do BERTopic: a visualização dos termos com o *score* de cada um em cada tópico é uma forma mais intuitiva e fácil de entender o relacionamento das palavras com seus tópicos e entre si.
- Utilize alguma outra técnica de visualização que você acha interessante.
- Em comparação com o modelo anterior, os tópicos são mais interpretáveis qualitativamente? O que significa o *score* associado a cada termo de um tópico em cada um dos casos?

Passo 5: Avaliando com uma métrica

Após avaliar qualitativamente seus tópicos, você deve avaliá-los quantitativamente. Para isso, você usará o NMPI (Normalized Pointwise Mutual Information), o qual avalia a coocorrência das palavras presente em um tópico nos mesmos documentos. Você pode ler mais sobre essa métrica em:

- https://dl.acm.org/doi/10.1145/3617023.3617040

Sistemas de Recomendação

Os Sistemas de Recomendação (SR) são uma área interdisciplinar que combina ciência da computação, machine learning, matemática, para estudar a interação entre sistemas e usuários. O objetivo principal dos SR é permitir que os sistemas compreendam as preferências dos usuários, permitindo a recomendação de itens personalizados com maior potencial de conversão.

Leitura recomendada

- Approaches, Issues and Challenges in Recommender Systems: A Systematic Review
- How good your recommender system is? A survey on evaluations in recommendation
- Recommender Systems: Issues, Challenges, and Research Opportunities
- Recommender Systems Handbook

Exercício 1 - Primeiro Recomendador

Esse exercício tem como objetivo te ensinar as técnicas básicas de Recomendação e mineração de dados. Aqui você verá um pouco como analisar os dados, identificar padrões, e no final a criação do primeiro recomendador, incluindo sua avaliação.

Passo 1: Análise dos dados

No geral as bases de dados para recomendação possuem três informações cruciais, o usuário, o item que ele interagiu e seu feedback, podendo ser uma nota, ou um feedback implícito, como um clique. Para começar, escolha uma das bases de dados abaixo para o exercício, a principal diferença entre elas é o cenário em que estão inseridas. Para fazer o download basta pesquisar por “<nome-dataset> recommendation dataset download” no google.

Datasets	Domain	Users	Items
MovieLens	Movies	138,493	26,744
Netflix	Movies	429,584	17,770
Yahoo Music	Music	10,000	13,214
LastFM	Music	5,000	36,718
Good Books	Books	53,424	10,000
Good Reads	Books	14,639	9,999
Amazon Store	Products	14,776	68,921
Clothing Fit	Clothes	105,508	5,850

Realize as seguintes análises:

1) Obtenha uma visão geral dos dados
    a. O número de usuários
    b. O número de itens
    c. A esparsidade do conjunto de dados

2) Obtenha informações dos usuários. Como a distribuição por idade, gênero, etc. Explorar dados adicionais sobre os usuários (quando existirem).

3) Obtenha informações dos itens. Como a distribuição por categorias, tags, etc. Caso já tenha feito o passo a passo de NLP, pode utilizar as técnicas de modelagem de tópico no caso de haver texto e descrições dos itens.

4) Criar algumas distribuições sobre a interação usuário-item:

    a. Plotar a frequência de avaliações
    b. Plotar a popularidade dos itens (ou seja, o número de usuários distintos que avaliaram cada item) — dica: ordenar a popularidade de forma decrescente
    c. Plotar o histórico de interações do usuário (ou seja, o número de itens avaliados por cada usuário) — dica: ordenar a distribuição de forma decrescente.
    d. Plotar o número de itens avaliados pelos usuários por um intervalo de tempo arbitrário (como o número de avaliações por dia/mês/ano).
    e. Plotar uma distribuição acumulada de usuários que entraram no sistema (você pode considerar que um usuário entrou no sistema ao obter o timestamp associado ao primeiro item avaliado por ele/ela).

5) Cruzar informações demográficas com as interações usuário-item

1. Plotar/imprimir o título dos itens mais populares no sistema
2. Plotar as categorias mais avaliadas pelos usuários
3. Dividir os usuários em grupos de idade e tentar realizar algumas análises para cada um deles

Passo 2: Recomendação Most Popular

Agora vamos fazer o primeiro sistema de recomendação. Nesse caso temos como o modelo inicial o Most Popular, que trata-se de recomendar os itens mais populares da base para cada usuário. Para esse cenário temos dois parâmetros principais, a quantidade itens a serem recomendados e se irá ou não repetir os itens que o usuário já tenha visto no histórico.

Basicamente, esse recomendador já foi criado no final das análises de dados, estruture melhor o código para ser utilizado como uma classe de recomendação.

Passo 3: Avaliação do algoritmo

Para avaliar a qualidade do recomendado, implemente as principais métricas, Precision, Recall, F1, NDCG, utilize como base o artigo Evaluating Recommender Systems. how to approach evaluating… | by Paul | Medium.

Passo 4: Exploração de novos algoritmos

Os sistemas de recomendação possuem três categorias principais, **collaborative-filtering**, o qual utiliza da similaridade entre usuários, ou similaridade entre itens, o famoso "usuários como você também viram", ou "itens semelhantes ao que você viu". **Content-based,** utiliza de metadados nos dados dos itens, como descrições, título, para adicionar mais informações ao contexto do item. O último é a junção desses dois modelos **Híbridos,** no qual juntamos collaborative-filtering e content-based para agregar ainda mais informações à recomendação. Abaixo segue materiais de apoio para implementação e entendimento desses modelos de recomendação.

Introduction To Recommender Systems- 1: Content-Based Filtering And Collaborative Filtering | by Abhijit Roy | Towards Data Science

How Recommender Systems Work (Netflix/Amazon)

How does Netflix recommend movies? Matrix Factorization

Exercício 2 - Explorando outros recomendadores

Esse exercício tem o objetivo de ajudar a fixar mais o conhecimento sobre recomendadores, implementação e funcionamento dos algoritmos de recomendação.

Passo 1: Implementação de algoritmos

Com o intuito de conhecer outros algoritmos e as suas diversas possibilidades de aplicação, recomendo implementar os seguintes algoritmos:

1. Matrix Factorization (e/ou Non-Negative Matrix Factorization)
2. KNN (K-Nearest Neighbors)
3. SVD (Singular Value Decomposition)

Obs: Os algoritmos 1 e 3 são de decomposição de matrizes, já o 2 é um algoritmo de clusterização.

Passo 2: Uso prático

Aplicar os conceitos de Content-Based e Collaborative-Filtering nos algoritmos implementados acima.

Passo 3: Avaliação dos algoritmos

Utilizando os conhecimentos adquiridos anteriormente, entenda e implemente as seguintes métricas e as compare para os algoritmos implementados:

1. F1-Score;
2. MSE;
3. RMSE.

Por fim, utilizar teste estatístico de Wilcoxon para comparar o ganho estatístico entre pares de algoritmos.

Elaborado por Fonseca , Reis, Prenassi, Yan, Ana