Trend-Desafios

Desafio 1: Tecnologia em Big Data

Jallysson Miranda Rocha

jallysson@usp.br

Objetivos

1. Avaliar se a Trend.Cards precisa usar a tecnologia big data.

2. Conhecer as opções de tecnologias atuais em big data.

3. Avaliar qual tecnologia em big data do mercado mais se encaixa com os requisitos e entregas da trend.cards.

Avaliar se a Trend.Cards precisa usar a tecnologia big data.

Sobre a Trend.Cards

A trend.cards é uma plataforma de acompanhamento de tendências. Com base em comportamento de mercado, simplificamos a grande massa de dados (Big Data) gerados na internet e redes sociais para facilitar a tomada de decisão relativa ao destino do seu negócio (**Trend.Cards**).

Para isso, é necessário a captura, análise e indexação de uma quantidade massiva de dados:

- textos;
- imagens;
- entre outros.

Sem o uso de tecnologias

Antes de verificar as tecnologias disponíveis, foram realizados testes sem o uso de tecnologias de *Big data*. Apenas, programação nativa com uso de módulos disponíveis na linguagem de programação *python*.

Módulos utilizados no ambiente de teste:

- ***twitter*** - acessa a *API* do *twitter* com autenticação de perfil;
- ***csv*** - manipula arquivos com a extensão .csv;
- ***math/numpy*** - indispensável em computação científica;
- ***re/unicodedata*** - usados para fazerem *tokenize* em dados textuais;
- ***matplotlib*** - muito usado em análise de dados, por meio de figuras;
- ***sklearn*** - muito utilizado em aprendizado de máquina, com algoritmos de agrupamento, classificação entre outros;
- ***threading*** - manipula *threads*.

Capturando os dados

Um exemplo de **captura de dados**, foi realizado com o *twitter* em tempo real (arquivo: *python-twitter.py*). No período de teste, foram coletados um pouco mais de 2100 *tweets* em intervalos de tempo definidos, executados por *threads* para tentar minimizar as limitações da API.

Também foram coletados as *trending topcs* mundiais e as do Brasil no momento de execução (arquivo: *trending-topics.csv*) e as imagens contidas nos *tweets*, que podem ser estudadas também para a formação de grupos.

Capturando os dados II

Limitações: **limite de *tweets*** por solicitação e **limite de tempo** entre solicitações. No teste, foi realizado solicitações de três em três minutos, buscando os últimos 20 *twittes* da *timeline*, e não houve falha na API nesse intervalo.

Para minimizar o problema de limite de *tweets* por solicitação, é realizado o armazenamento desses dados a cada solicitação, sem realizar duplicadas (arquivo: *dataset-twitter.csv*). Fazendo uso desse arquivo para experimentação em algoritmos de aprendizado de máquina. (Existem diversas maneiras de resolver esses problemas).

Pré-processando os dados

Entre os dados coletados de um *tweet*, foram usados apenas as mensagens (texto). E realizado a análise léxica (arquivo: *tokenize.py*) removendo caracteres especiais e acentuações para poder trabalhar com os dados (maneira mais simples).

Para trabalhar com dados textuais em algoritmos de aprendizado de máquina, eles precisam ser representados adequadamente. Existem diversas técnicas, como:

- binária (os dados tem a mesma importância);
- frequência de cada termo no texto (cada termo tem um peso específico);
- frequência do termo no texto pela frequência inversa do termo em todos os textos (TF-IDF) - técnica usada no teste (arquivo: *tf_idf.py*);
- entre outras.

Cada texto do *tweet* passa a ser um vetor, em que cada palavra é um atributo (arquivo: *encoding-twitter.csv*).

Para o experimento, é realizado a representação TF-IDF, e a formação de grupos de termos e *tweets* com o algoritmo de agrupamento *k-means* do módulo *sklearn*. Um exemplo do teste é mostrado na figura 1, os grupos são identificados pelas cores:

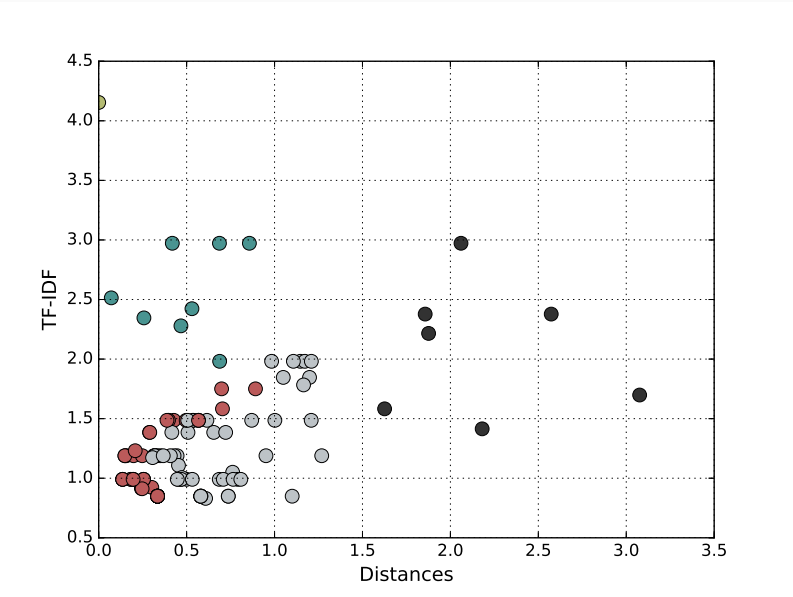

Figure 1: Cinco grupos de *tweets* formados pelo algoritmo *k-means*.

Experimento de teste

Além da análise dos resultados por gráficos (arquivo: *plot.py*), também é gerado um arquivo (*clusters-twitters.csv*) de saída com os termos dos *tweets* e seus respectivos grupos.

Table 1: Exemplo de um resultado esperado no arquivo de saída.

Cluster	*Term1*	*Term2*	*Term3*	..	*TermN*
2	jogo	futebol	foxsport		
1	novo	iphone	câmera		
2	corinthians	ganha	jogo	..	arena
:	:	:	:	:	
1	vendo	celular	novo		

A Trend precisa de tecnologia de Big Data?

Para coletar, experimentar e analisar os dados não a necessidade de tecnologias de *Big Data*. Módulos em *python* conseguem realizar esse trabalho (mesmo que as tecnologias também já forneçam esses serviços prontos). Podem ser usados serviços na nuvem para melhorar o tempo de execução (ex: *Azure* da *Microsoft*), com *hardwares* melhores e com maior capacidade de armazenamento.

Minha sugestão, é focar no pré-processamento dos dados e estudar o seu uso em algoritmos supervisionados e não supervisionados de aprendizado de máquina com implementações próprias. Assim, a *Trend* se preocupa com a **qualidade dos dados** e dos **resultados**, e não apenas a armazenar uma enorme quantidade de dados e processar todas elas ao mesmo tempo.

Além disso, algoritmos podem encontrar padrões nos dados fornecidos de treinamento, e não havendo necessidade de uma quantidade massiva de dados.

Conhecer as opções de tecnologias atuais em big data.

Principais:

- *Hadoop* - permite o processamento distribuído de grandes conjuntos de dados estruturados, semiestruturados e não estruturados. IBM [3];
- *Spark* - foco em velocidade, facilidade de uso e análises sofisticadas. Penchikala [6];
- *Storm* - foco em tolerância a falhas e gerenciamento. Tim [4];
- *DataTorrent RTS* - foca em *streaming* e análise em tempo real capaz de processar mais de um bilhão de eventos por segundo. Abel [1];
- *Samza* - processador de fluxo de (*stream processing*) em tempo real. Mauro [5];
- *Flink* - plataforma para processamento de dados de forma eficiente, distribuída e de uso geral. Jean [2].

A *Spark* pode ser uma alternativa de uso com o *python*

Ambas as tecnologias oferecem capacidade de rápido processamento de grandes quantidades de dados. Mas, conhecendo um pouco dessa área, é preciso tomar cuidado em generalizar os dados (dados do *twitter* e do *facebook* podem ter significados diferentes). É preciso conhecer o tipo dos dados e aplicar o pré-processamento adequado.

Avaliar qual tecnologia em big data do mercado mais se encaixa com os requisitos e entregas da trend.cards.

Tecnologia

Como já mencionado, usando a linguagem *python* com a importação de módulos é capaz de satisfazer as necessidades da *trend*. Como, coletar dados (textuais, figuras entre outras) e fazer análises de dados. Então na minha opinião, se trabalhar sem tecnologias, apenas com a linguagem *python*, que é muito usada em *big data* e aprendizado de máquina, é a melhor opção de oferecer um melhor tratamento aos dados e contribuir com resultados promissores, sendo capaz de desenvolver novas técnicas nessa área para tratamentos de dados de *big data*.

Referências I

A. Avram and P. V. Rendeiro.

Datatorrent 1.0: gerenciando mais de 1 bilhão de eventos por segundo em tempo real.

Ano de visualização (2017), disponível em: `https://www.infoq.com/br/news/2015/04/datatorrent`, 2015.

J. Bez.

Plataformas de big data: Spark, storm e flink.

Ano de visualização (2017), year =.

IBM.

O que é o hadoop?

Referências II

T. M. Jones.

Processe big data em tempo real com twitter storm.

Ano de visualização (2017), disponível em: `https://www.ibm.com/developerworks/br/library/os-twitterstorm/`,
2012.

M. Nogueira.

Uma introdução ao apache samza.

Ano de visualização (2017), disponível em: `https://www.linkedin.com/pulse/uma-introdução-ao-apache-samza-mauro-alexandre`,
2016.

S. Penchikala and L. Santana.

Big data com apache spark - parte 1: Introdução.

Ano de visualização (2017), disponível em: `https://www.infoq.com/br/articles/apache-spark-introduction`,
2015.

Trend-Desafios

Desafio 1: Tecnologia em Big Data

Jallysson Miranda Rocha

jallysson@usp.br