Machine Learning Engineer Nanodegree

Capstone Project

Júnio de Freitas

Agosto de 2017

Usando Machine Learning para Identificar Perguntas Similares do Quora

---

I. Definição

Visão Geral do Projeto

Identificar se um par de perguntas refere-se ao mesmo assunto não é trivial. Partindo dessa premissa foi que o site Quora, uma forma de rede social de compartilhamento de conhecimento através de perguntas e respostas sobre qualquer coisa, disponibilizou um dataset e patrocinou uma competição na plataforma Kaggle com o objetivo de classificar como iguais ou diferentes um conjunto de perguntas, extraídas de sua plataforma, usando Machine Learning. Com esse objetivo, este projeto visa utilizar funções de similaridade textuais junto com um hiper-classificador, um comitê, formado por classificadores para tentar identificar os pares de perguntas corretamente.

Definição do Problema

O problema de identificar se duas perguntas distintas se referem ao mesmo assunto envolve descobrir se o contexto de cada pergunta é o mesmo. Foi para isso que a plataforma Quora disponibilizou um dataset formado de pares de perguntas extraídas de lá e patrocinou uma competição no Kaggle[1] para competidores solucionarem esse problema usando as mais variadas técnicas de Machine Learning. O Quora recebe mais de 100 milhões de visitas ao mês e, comumente são feitas perguntas similares. Tal situação traz problemas a plataforma, como maior tempo para encontrar a melhor resposta ou que se responda a mesma pergunta mais de uma vez ou até redirecionando o usuário para uma pergunta com as respostas similares a que ele iria fazer, de forma a melhorar a experiência de navegação do usuário pela plataforma. O interesse de se resolver esse problema recai na complexidade da área de Processamento de Linguagem Natural[2], mais especificamente na aplicação de técnicas que identificam textos similares baseados na semântica [3]. Entretanto, este projeto utilizará técnicas baseadas em similaridade textual provindas da área de Mineração de Dados e aplicadas para identificar registros duplicados [4]. Essas técnicas serão descritas em detalhes na seção Técnicas e Algoritmos. Dentro desse contexto, para lidar com o problema de classificar os pares como verdadeiros ou falsos, problema de classificação binária, serão usadas técnicas de Machine Learning (Aprendizado de Máquina), mais especificamente Aprendizado Supervisionado, onde cada classificador precisa treinar, com exemplos de treino, para aprender a classificar os exemplos de teste.

Métricas

As métricas utilizadas nesse projeto são as tradicionalmente usadas para avaliar a qualidade de classificadores na área de Mineração de Dados e Recuperação de Informação. Por outro lado, identificar se pares de registros são duplicados requer algoritmos de classificação binária dos quais devem ser aplicadas métricas de avaliação que não apenas contabilize acertos, mas que classifiquem corretamente cada classe. Medir a habilidade de um algoritmo em corretamente identificar entidades equivalentes é a melhor forma de avaliar a qualidade dos resultados dos experimentos.

Neste projeto as métricas utilizadas foram:

- **Precisão (Precision)**: é a razão entre o número de itens classificados corretamente como verdadeiros e o número total de itens classificados como verdadeiros. Um item classificado corretamente como verdadeiro é denominado *verdadeiro positivo*. Um item classificado incorretamente como verdadeiro é denominado *falso positivo*.

$$Precision = \frac{verdadeiros\ positivos}{verdadeiros\ positivos + falsos\ positivos}$$

- **Revocação (Recall)**: é a razão entre o número de itens classificados corretamente como verdadeiros e o número total de itens verdadeiros. O total de itens verdadeiros é a soma dos verdadeiros positivos com os falsos negativos. Um item classificado incorretamente como falso é denominado *falso negativo*.

$$Recall = \frac{verdadeiros\ positivos}{verdadeiros\ positivos + falsos\ negativos}$$

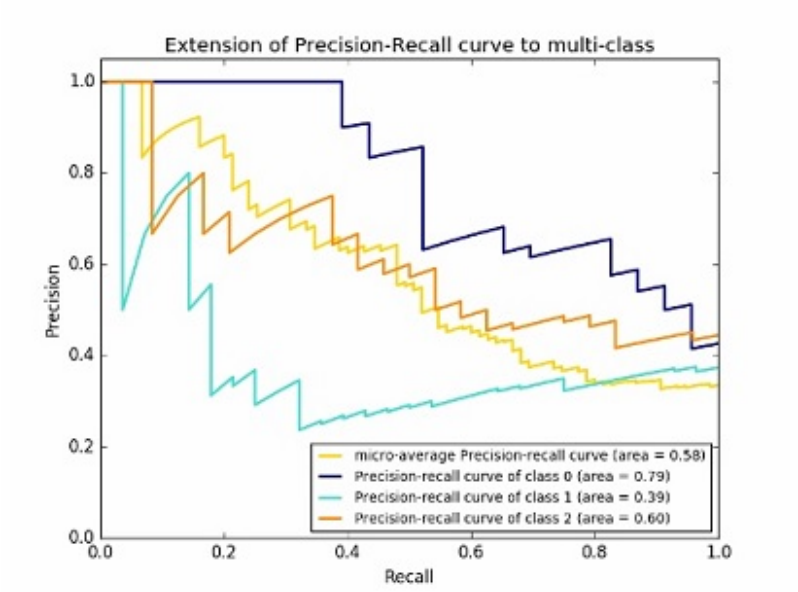
- **Medida F (F1)**: é a média harmônica entre a *precisão* e a *revocação* que produz uma noção mais precisa da acurácia de um classificador. Essa é a métrica mais significativa utilizada neste projeto, pois dá a noção exata se os pares estão sendo classificados corretamente como verdadeiros ou falso, independente de como estejam distribuídos no dataset. Quando mais próximo de 1 for F1 mais perfeita será a classificação de um determinado classificador.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- **Acurácia (Accuracy)**: é a razão entre o número total de itens classificados corretamente e o número total de itens.

$$Accuracy = \frac{Total\ acerto}{Total\ itens}$$

- **Curva Precisão-Revocação (Precision-Racall Curve)**: é um gráfico de uma curva que relaciona os valores de *precisão* e *revocação* para diferentes limiares (*thresholds*). Essa curva é usada principalmente para avaliar um classificador binário. No gráfico o eixo *x* é a revocação e o eixo *y* a precisão. O gráfico abaixo mostra um exemplo dessa curva [5].



- **Precisão Média (Average Precision)**: é a precisão média dos escores dos classificadores. Corresponde a área abaixo da curva de precisão-revocação.

- **Matriz de Confusão (Confusion Matrix)**: é uma matriz ou tabela que permite visualizar o desempenho de um classificador. Cada coluna da matriz representa a classe predita pelo classificador e cada linha presenta a real classe, ou vice-e-versa. Para uma classificação binária, a matriz $2X2$ será composta pelo número de verdadeiros negativos, falsos negativos, verdadeiros positivos e falsos positivos, conforme pode se observar na figura abaixo [6].

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

- **Perda Logística(Logistic Loss)**: também conhecido como Log Loss ou Cross-entropy Loss, é uma função de perda [7] comumente usada em *regressão logística* e suas extensões tal como as *redes neurais*. Definida como a probabilidade logarítmica negativa dos itens classificados como verdadeiros dado as predições de um classificador probabilístico.

$$Log\ Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

II. Análise

Análise dos Dados

A plataforma Quora disponibilizou dois datasets, o *train* e *test* datasets para a competição do Kaggle, um para treino e um para teste, respectivamente. Entretanto, somente o *train* dataset está rotulado, isto é, contém a classe correta para cada item do dataset. Dessa forma, neste projeto será utilizado somente o dataset *train*. Os rótulos foram fornecidos manualmente por especialistas e, é certo que pode ter havido ruídos nesse processo, ou seja, o dataset de treino não garante que seus dados foram rotulados de forma 100% precisa.

O objetivo da competição do Quora (*Quora Questions Pairs*) é prever quais dos pares de perguntas fornecidos contêm duas questões com o mesmo significado. Cada item do dataset é composto conforme tabela abaixo:

Campo	Tipo	Descrição
id	Inteiro	identificador único de um par de questões do dataset (chave primária)
qid1	Inteiro	identificador único da questão #1 do par
qid2	Inteiro	identificador único da questão #2 do par
question1	Texto	Texto completo da questão #1 do par
question2	Texto	Texto completo da questão #2 do par
is_duplicate	Inteiro	rótulo do item, é definido como 1 se o par de questão 1 e questão 2 tiverem essencialmente o mesmo significado e 0, caso contrário. Classe binária.

A tabela abaixo mostra algumas estatísticas desse dataset:

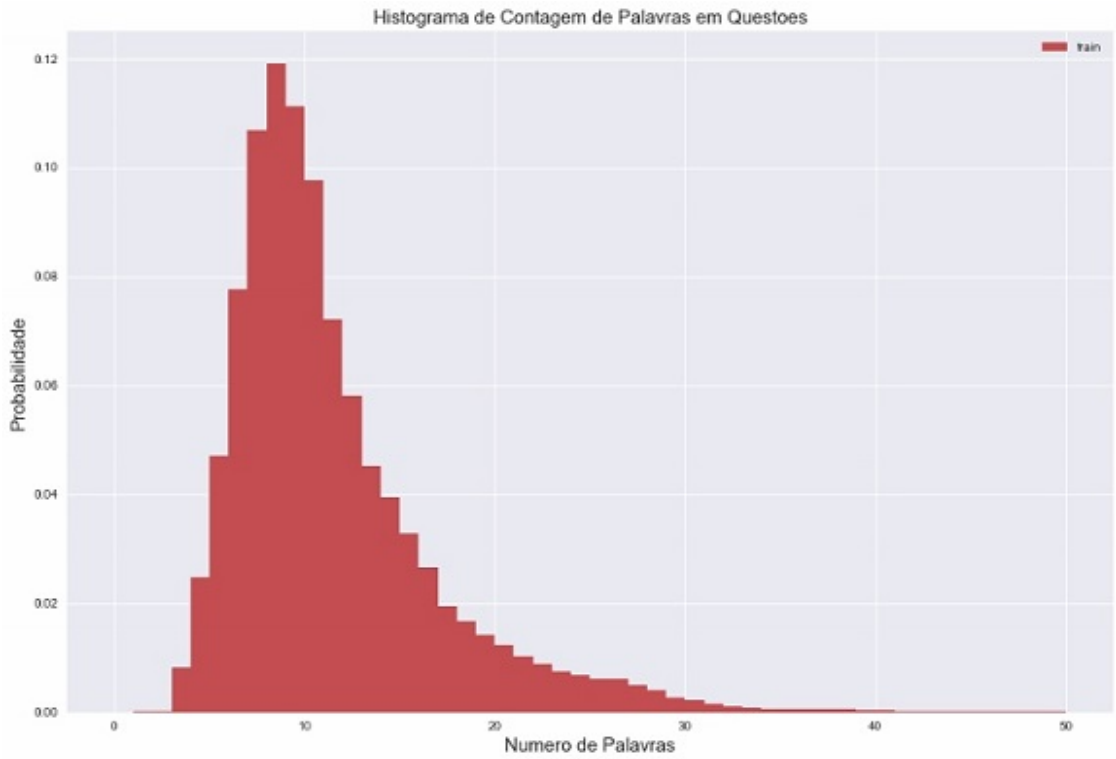
Nome	Valor
Tamanho	`63.4MB`
Total Itens	`404.290`
% Pares Duplicados	`37%`
Total Pares Duplicados	`149263`
Total Pares Não Duplicados	`255027`
Total de Questões	`537933`
Toal de Questões que Aparecem em Mais de um Par	`111780`
Total Tokens	`2797439`
Total Tokens Distintos (Vocabulário)	`85092`

A seguir é mostrada uma amostra dos dados com pares de registros duplicados e não duplicados do *train* dataset:

id	qid1	qid2	question1	question2	is_duplicate
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
11	23	24	How do I read and find my YouTube comments?	How can I see all my Youtube comments?	1

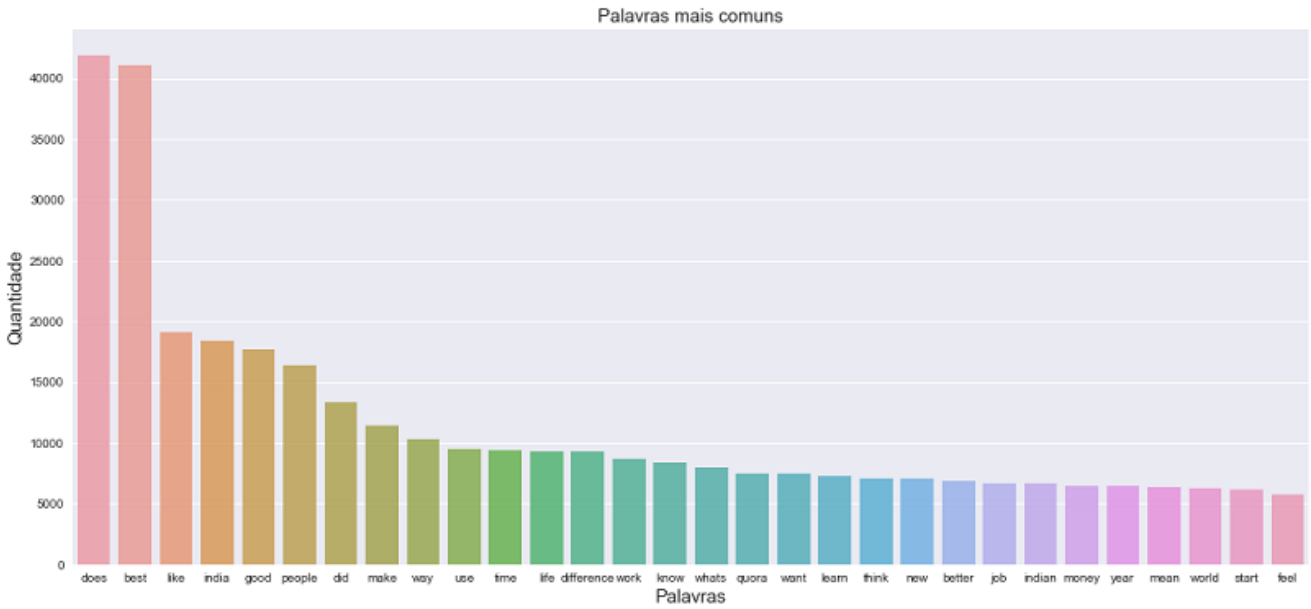
Visualização Exploratória

Para compreender melhor o *train* dataset foram realizadas algumas análises visuais. Primeiramente, é mostrado um histograma que conta o número de palavras por questões.



O gráfico "Histograma de Contagem de Palavras em Questões" mostra a distribuição da quantidade de palavras nas questões do *train* dataset. É possível verificar que a maioria das questões possuem aproximadamente 10 palavras, com essa informação é possível estimar o que usar para processar os pares de questões.

O próximo gráfico mostra as palavras (*tokens*) que mais aparecem nas questões do dataset.



Este último gráfico mostra que as palavras que mais ocorrem, por exemplo, *"does","did","whats"* são stopword [8]. Com isso, é possível realizar um pré-processamento do dataset e remover as palavras stopword para avaliar se haverá uma maior qualidade. Nos experimentos realizados são apresentados os resultados do dataset processado com e sem stopword.

Técnicas e Algoritmos

Neste projeto foram usadas técnicas de deduplicação para tentar identificar as questões do dataset do Quora. O problema de deduplicação está diretamente relacionada à teoria de *Record Linkage* (RL) [9] que tem como objetivo encontrar réplicas de registros em banco de dados (ou datasets). A diferença é que RL é aplicada no cruzamento de registros de dois ou mais banco de dados, enquanto que deduplicação é aplicada em um único banco de dados. Entretanto, os conceitos aplicados à deduplicação são idênticos aos de RL de tal forma que as principais soluções para o problema de deduplicação são produtos do desenvolvimento de anos em pesquisas em RL.

O elemento principal de que trata o problema de deduplicação é o *registro* ou *item*. Um registro é uma unidade de um banco de dados que representa uma entidade do mundo real, composta de um ou mais atributos que o caracterizam. Os registros podem ser comparados usando os valores de seus atributos tomados separadamente ou em conjunto, como sendo um único valor. Neste caso, trata-se do que é chamado de *análise sintática* de registros.

A maneira mais direta de identificar registros duplicados em um banco de dados é através da comparação de pares de registros. Dado um banco de dados A, um par de registro é definido como (a, b) onde $(a, b) \in A \times A$. Assim, sendo $E(a)$ e $E(b)$ as entidades que os registros a e b representam no mundo real respectivamente, um modelo de decisão deve classificar o par (a, b) como:

- par *verdadeiro*, se $E(a) = E(b)$;
- par *falso*, se $E(a) \neq E(b)$;
- par *indefinido*, quando o modelo não consegue definir uma classificação segura.

Através do modelo de decisão, o objetivo do processo de deduplicação é encontrar um conjunto de pares $T \subseteq A \times A$ onde $T = \{(a, b) | E(a) = E(b) \wedge (a, b) \in A \times A\}$ e um conjunto de pares $F \subseteq A \times A$ onde $F = \{(a, b) | E(a) \neq E(b) \wedge (a, b) \in A \times A\}$.

Os conjuntos de pares T e F formados após o processo de deduplicação devem ser classificados conforme rótulo original dos pares. Entretanto, em certas ocasiões a classificação é feita incorretamente, formando dois conjuntos definidos como:

- conjunto dos *Falsos Positivos* $= \{(a,b)|E(a) \neq E(b) \wedge (a,b) \in T\}$;
- conjunto dos *Falsos Negativos* $= \{(a,b)|E(a) = E(b) \wedge (a,b) \in F\}$.

Neste projeto um modelo de decisão é um algoritmo de classificação como *SVM*, *Decision Tree*, *Naive Bayes*, entre outros. Esses algoritmos que irão classificar o conjunto de pares em verdadeiros ou falsos e depois, será possível calcular as métricas de qualidade para comparar quais são os melhores.

Resumidamente, o processo de deduplicação é composto das seguintes etapas:

1. Pré-procesamento, limpeza e normalização dos dados do dataset;
2. Blocagem e geração dos pares dos dados do dataset;
3. Cálculo da similaridade entre os pares de questões do dataset;
4. Classificação dos pares baseado nos vetores de similaridade (*features*);
5. Avaliação dos resultados.

Pré-procesamento, limpeza e normalização dos dados do dataset

O primeiro passo consiste em tratar os dados do dataset e prepará-los para serem processados nas etapas seguintes. Esse tratamento consiste em transformar, normalizar, remover ruídos, formatar os dados, etc. Os detalhes desse processo serão tratados na próxima seção.

Blocagem e geração dos pares dos dados do dataset

O segundo passo é resume-se numa técnica, bastante utilizada na área de Mineração de Dados [10], que consiste em dividir o dataset em blocos, onde cada bloco é composto por registros que tenham algo em comum, como por exemplo, alguma palavra em comum (*tokens* em comum). A ideia da blocagem é reduzir o número de pares gerados para serem comparados, porque na maioria dos casos de deduplicação o número de pares verdadeiros é a minoria, normalmente por volta de 1%. Com os blocos do dataset criados é possível gerar conjuntos de pares por blocos. Assim, define-se um dataset como $A = B_1 \cup B_2 \cup \ldots \cup B_n$ para $n \geq 2$. Cada subconjunto B_i é chamado de bloco. O que define quais os registros irão pertencem a um determinado bloco são as regras ou esquema. Uma regra *x* é uma função que mapeia todos os registros que concordam com ela ao bloco B_x. Exemplos de regras de blocagem podem ser: "nomes que começam com a letra a" ou "3 primeiras letras em comum do sobrenome". Neste projeto será usada a blocagem tokens em comum.

Cálculo da similaridade entre os pares de questões do dataset

O passo seguinte é processar o conjunto de pares gerados utilizando funções de similaridade de texto ou métricas de similaridade[11]. Tradicionalmente, operações de consultas em banco de dados são baseadas na igualdade de valores. Quando se fala em casamento aproximado ou em busca aproximada já não é mais possível realizar tais operações. Para solucionar este problema a literatura apresenta diversas funções de similaridades de domínio específico que também são conhecidas como *métricas de similaridade*. Há métricas de similaridade para vários tipos de objetos, como textos, sons, imagens, mapas, séries temporais, cadeias de DNA, etc. Independente disso, uma métrica de similaridade f é definida como:

$$f(a,b) = v$$

Onde $a, b \in D$, D é o domínio dos objetos e $v \in \mathbb{R}$ é o valor real que representa a similaridade entre os objetos. Normalmente o valor de v é normalizado de tal forma que $v \in [0, 1.0]$. Em relação à deduplicação, assume-se que o domínio é textual e os objetos são cadeias de caracteres (*strings*).

As métricas textuais estão divididas de acordo com as características dos textos no qual são aplicadas. Além disso, as métricas podem ser divididas de acordo com a natureza semântica dos dados que formam os registros, como definido abaixo:

- **Métricas de caracteres**: são métricas que calculam a similaridade entre duas strings a partir dos caracteres que as formam. Devem ser aplicadas de preferência em uma única palavra ou frases curtas, por exemplo nomes, sobrenomes, cidades, etc;
- **Métricas de token**: são métricas que comparam duas strings, ou seja, palavra por palavra. Quanto mais as palavras comparadas forem iguais, maior será a similaridade entre as strings. Essas métricas são ideais na comparação de strings longas como endereços, lista de autores, parágrafos de textos, etc;
- **Métricas fonéticas**: são métricas que comparam codificações fonéticas entre duas strings de tamanho curto. Essas métricas dependem dos algoritmos que transformam uma string em um código fonético e a partir desses códigos as strings são comparadas. Normalmente essas métricas variam de idioma para idioma;
- **Métricas híbridas**: são métricas que calculam a similaridade entre duas strings tanto pelos caracteres quanto pelas palavras que as formam, ou seja, misturam numa mesma métrica, as de caracteres e as de token. São métricas ideais para serem aplicadas sobre pares de questões do Quora.

Para as definições abaixo, considerar as strings A e B, que serão chamadas de *Bag*, como conjuntos formados pelas palavras a e b, onde cada palavra é associada a um peso w, respectivamente.

- **Levenstein**[10]: também conhecida como Distância de Edição , calcula o menor número de operações de edição para transformar uma string na outra. As operações de edição são: cópia, inserção, substituição e exclusão. Todas essas operações têm peso 1, exceto a operação de cópia, que te peso 0. A similaridade é calculada como:

$$Levenstein(a, b) = D(a_i, b_j) \text{ e}$$
$$D(a_i, b_j) = min \begin{cases} D(a_i - 1, b_j - 1) + d(a_i, b_j) \\ D(a_i - 1, b_j) + 1 \\ D(a_i, b_j - 1) + 1 \end{cases}$$

Onde D é a matriz de custo entre os caracteres das strings a e b, e $d(a_i, b_j) = \begin{cases} 1, & \text{se } a_i = b_j \\ 0, & \text{caso contrário} \end{cases}$.

- **Jaro**[12]: Dado duas string a e b, considerar a' como os caracteres em a comuns em b e b' os caracteres em b comuns em a. Para ser comum os caracteres têm que ser iguais e estar na mesma posição. Assim, a similaridade é calculada como:

$$Jaro(a, b) = \frac{1}{3} * \left(\frac{|a'|}{|a|} + \frac{|b'|}{|b|} + \frac{|a'| - T(a', b')}{2 * |a'|} \right)$$

Onde $T(a', b')$ é o número de caracteres em a' iguais a b' mas que estão em diferentes posições.

- **Jaro-Winkler**[13]: É uma variante da métrica Jaro que usa um comprimento P indicando o maior prefixo comum entre as strings comparadas. Dadas as strings a e b e $P' = max(P, 4)$, a similaridade é calculada como:

$$JaroWinkler(a, b) = Jaro(a, b) + \frac{P'}{10} * (1 - Jaro(a, b))$$

Essa métrica corrige o problema de comparar uma string grande com uma string pequena, neste caso, o prefixo P de tamanho 4 foi considerado ideal.

- **Jaccard**[14]: É uma métrica que mede o número de palavras em comum entre duas strings independentes da ordem. A similaridade é calculada como:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

É bastante recomendada na comparação entre strings longas, como por exemplo parágrafos de textos.

- **TFIDF**[14]: é uma métrica que se baseia em vetores que são formados a partir do produto $TF * IDF$ de cada palavra do texto. O TF é a freqüência de cada palavra no registro e o IDF é a freqüência inversa do número de registros que contém determinada palavra. Essa métrica também é chamada de $Cosseno$ e é bastante utilizada na comunidade de Recuperação de Informação [15]. A similaridade é calculada como:

$$TFIDF(A, B) = \sum_{t \in A \cap B} V(t, A) * V(t, B)$$

Onde $V(t, A) = \frac{V'(t,A)}{\sqrt{\sum_{t \in A} V'(t,A)^2}}$, $V'(t, A) = \log(TF_{t,A} + 1) * \log(IDF_t)$, $TF_{t,A}$ é a frequencia da palavra t na string A e IDF_t é o IDF da palavra t.

- **SoftTFIDF**[15]: Versão flexível da métrica TFIDF, onde palavras similares acrescentam peso ao resultado, ou seja, não só compara o peso das palavras em comum entre as strings, mas também utiliza uma métrica de caractere para encontrar palavras parecidas. Sendo $CLOSE(l, A, B)$ o conjunto de palavras $a \in A$ e $b \in B$ tal que $f(a, b) > l$, onde f é uma métrica de caractere, a similaridade é calculada como:

$$SoftTFIDF(A, B) = \sum_{t \in CLOSE(l,A,B)} V(t, A) * V(t, B) * D(t, B)$$

Onde $D(t, B) = max_{b \in B} f(t, b)$.

- **Soundex**[16]: É uma métrica que compara strings pelos sons que produzem. A métrica gera um código de 4 caracteres para cada string que representa sua pronúncia, esse código é composto por uma letra seguida de três dígitos entre 0 e 6. Essa métrica encontra nomes que possuem pronúncia similar. É calculada como:

$$Soundex(a, b) = func(soundexcod(a), soundexcod(b))$$

Onde $func$ é uma métrica de caractere qualquer e $soundexcod(a) = a_1 C(a_2) C(a_3) C(a_4)[\dots]$. $C(a_i)$ retorna a chave *soundex* de acordo com o caractere a_i, tal como:

$$C(a_i) = \begin{cases} 1, & \text{se } a_i \in \{B, F, P, V\} \\ 2, & \text{se } a_i \in \{C, G, J, K, Q, S, X, Z\} \\ 3, & \text{se } a_i \in \{D, T\} \\ 4, & \text{se } a_i \in \{L\} \\ 5, & \text{se } a_i \in \{M, N\} \\ 6, & \text{se } a_i \in \{R\} \\ 0, & \text{caso contrário} \end{cases}$$

- **MongeElkan** [17]: esta métrica foi proposta por Monge e Elkan baseada na métrica original de Smith-Waterman [18]. Desenvolvido para ser aplicado nos problemas de alinhamento de DNA, é calculada como:

$$MongeElkan(a,b) = \sum_{i}^{max(|a|,|b|)} \begin{cases} -3, & \text{se } a_i \neq b_i \\ +5, & \text{se } a_i = b_i \\ +3, & \text{se } a_i \approx b_i \end{cases}$$

Onde dois a_i e b_i caracteres de cada palavra são aproximados quando são $\{dt\}$ $\{gj\}$ $\{lr\}$ $\{mn\}$ $\{bpv\}\{aeiou\}$ {,.}

- **Smith-Waterman**[18]: é um algoritmo de alinhamento sequencial que determina regiões similares entre duas sequencias de proteínas (DNA). A similaridade é calculada como:

$$SmithWaterman(a,b) = T(a_i, b_j) \text{ e}$$
$$T(a_i, b_j) = max \begin{cases} T(a_{i-1}, b_{j-1}) + \sigma(Sa_i, Sb_j) \\ T(a_{i-1}, b_j) + \text{gap penality} \\ T(a_i, b_{j-1}) + \text{gap penality} \\ 0 \end{cases}$$

Onde $T(a_i, b_j)$ é a matriz de caracteres das palavras a e b, linha e coluna, respectivamente. Inicialmente, a coluna e linha zero são preenchidas com zero.

- **BagSim**: essa métrica foi proposta exclusivamente para lidar com os dados do *train* dataset, pois se baseia na diferenças entre os pesos, $W(t)$, normalmente o IDF, de duas palavras entre duas strings. O cálculo se resume em:

$$BagSim(A,B) = \sum_{a \in A, b \in B | a=b \vee |W(a)-W(b)|<=lim}^{max(|A|,|B|)} W(a) * W(b)$$

- **FullSimilarity**: essa métrica também foi proposta para este trabalho. Ela computa a similaridade total entre duas strings. Dadas as strings A e B, se $a \in A$ e $b \in B$ e |A| e |B|, o número de palavras de A e B, respectivamente e $func$ uma métrica de caractere, a similaridade é calculada como:

$$FullSimilarity(A,B) = \frac{\sum_{a_i \in A \wedge b_i \in B} func(a_i, b_j)}{max(|A|, |B|)}$$

Classificar os pares baseado nos vetores de similaridade

O passo seguinte é calcular a similaridade entre os pares do conjunto de questões através das métricas descritas anteriormente. Ao final, do conjunto de pares será gerado um conjunto de vetores de similaridade (as *features*). Dado um par $p =< q_1, q_2 >$, formado pelas questões q_1 e q_2, um conjunto de métricas de similaridade M formado por n métricas de texto, vetor de similaridade será calculado como:

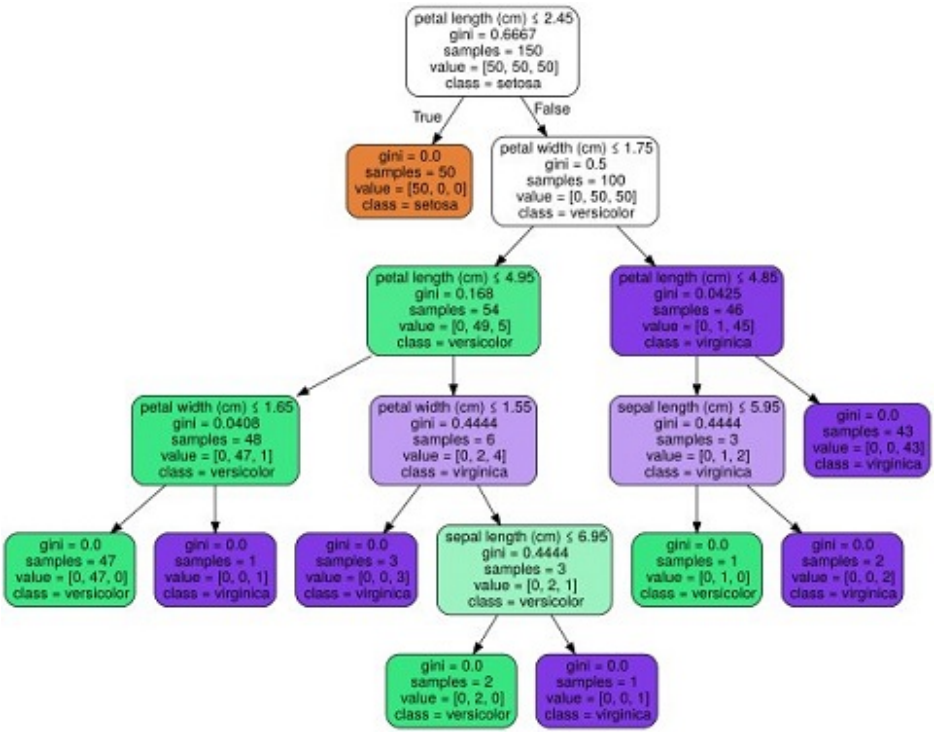
$$v_i = m_i(p) \; \forall m_i \in M$$

E para todo $v_i \in [0,1]$, assim um vetor similaridade pode ser representado como $V = [v_1, v_2, \ldots, v_n]$.

Esses vetores serão as *features* que os classificadores utilizarão para tentar identificar se os pares de perguntas são verdadeiros ou não.

Os classificadores que serão utilizados nesse projeto são:

- **Árvore de Decisão (DTree)**[19]: é um método de aprendizagem supervisionada não paramétrico usado para classificação e regressão. O objetivo é criar um modelo que prediga o valor de uma variável de destino aprendendo simples regras de decisão inferidas a partir dos dados das *features*. Na figura abaixo é mostrada uma representação gráfica de uma árvore de decisão, onde em cada nó há uma regra de decisão e dependendo do valor final dessa regra seguir-se-á ao nó filho adequado. Esse processo é repetido até se chegar nos nós folhas, aqueles que não possuem nós filhos.



- **Gaussian Naive Bayes (GaussianNB)**[20]: esse classificador implementa o algoritmo Gaussian Naive Bayes para realizar classificação. A probabilidade das *features* calculadas como gaussianas é:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Os parâmetros σ_y e μ_y são estimados usando a máxima probabilidade.

- **Multinomial Naive Bayes (MultinomialNB)** [21]: é um classificador Naive Bayes para modelos multinomiais. No campo da estatística uma distribuição multinomial é uma generalização da distribuição binomial. Esse classificador normalmente é empregado quando se tem *features* discretas (por exemplo em contagem de palavras para classificação de texto). A distribuição é parametrizada pelos vetores $\theta_y = (\theta_y 1, \ldots, \theta_{yn})$ para classe y, onde n é o número de *features* e θ_{yi} é a probabilidade $P(x_i|y)$ da feature i de uma amostra pertencente a classe y.
- **Stochastic Gradient Descent (SGD)**[22]: é um classificador linear, assim como SVM, Regressão Logística e outros, que usa no treinamento Stochastic Gradient Descent (Descida do Gradiente Estocástica). A perda do gradiente é estimada para cada amostra em um tempo e o modelo é atualizado ao longo do caminho com uma taxa de aprendizagem. Somente recentemente que o SGD vem sendo usado abundantemente principalmente no contexto de aprendizagem de larga escala, neste caso em classificação de texto e processamento de linguagem natural.
- **Multi-layer Perceptron classifier (MLP)**[23]: é uma implementaçao de redes neurais para classificação. Este modelo otimiza a função de perda de log usando LBFGS ou descida do gradiente estocástica.

- **Random Forest (RForest)**[24]: é um tipo de classificador formado por outros classificadores, normalmente por um conjunto de árvores de decisão. A classificação se baseia na média das estimativas do conjunto das árvores. O algoritmo cria as árvores aleatoriamente a partir de semi-treinos com partes do dataset. Uma vez que as árvores são geradas, somente uma pequena quantidade será útil para a classificação final, ou seja, muitas são descartadas ou suas estimativas não são levadas em conta. No processo de aprendizado, cada árvore gerada estima seu rótulo e, ao final é obtido a média de votação de todas as árvores como a previsão final.
- **Gradient Boosting (GBoosting)**[25]: assim como Random Forest o Gradient Boosting é um hiper classificador formado por um conjunto de outros classificadores, normalmente árvores de decisão. Neste caso, o GB calcula o erro (usando uma função de perda) para cada árvore conforme o algoritmo de Descida do Gradiente. É amplamente utilizado em competições e na resolução de problemas de classificação e regressão e é um dos métodos que obtém os melhores resultados.

Avaliar os resultados

A última etapa consiste na avaliação dos resultados obtidos pelos classificadores, depois que eles são treinados. O *train* dataset foi divido aleatoriamente treino e teste. Na etapa de avaliação os classificadores estimam o dataset de teste e métricas de avalição são aplicadas para verificar a qualidade que cada um atingiu. As métricas de avaliação foram descritas na seção **Métricas**. Com os resultados é possível comparar os modelos para ver qual o que se comporta melhor. Para avaliar os resultados é muito importante gerar gráficos para dar uma melhor ideia de como o modelo funciona.

Linha de Base

Como linha de base este projeto usará algoritmos de classificação supervisionada explicados na seção anterior: DTree, GaussianNB, MultinomialNB, SGD, MLP, RForest e GBoosting. Individualmente esses modelos lidam bem com classificação de texto binárias e alcançam bons resultados. Em especial, há diversos trabalhos na literatura que mostram que o RForest alcança ótimos resultados em problemas de identificação de registros duplicados [26].

III. Metodologia

Pré-processamento de Dados

O pré-processamento utilizado na base do Quora se resumiu em:

- Transformar todos os caracteres para minúsculo;
- Remoção de símbolos que não fossem letras e números, com exceção de alguns símbolos como "-";
- Remoção de espaços desnecessários entre as palavras;
- Preenchimento de campos nulos com o valor padrão `null`.

Outro pré-processamento importante é o que é chamado de *tokenização*, que é o processo de se obter os *tokens* (palavras) dos textos das perguntas, ou seja, a partir de um texto se obter uma lista de palavras distintas que formam esse texto.

A lista de token obtida pelo processo de *tokenização* é comumente chamada de 'Bag of Token' e armazena um outro valor associado a cada token, normalmente a frequência desse token no texto.

Com a lista de tokens é possível remover os tokens que podem não agregar informação na comparação entre os pares de questões do *train* dataset, nesse caso é aplicada a remoção das *stopwords*, que são palavras comuns de um vocabulário, normalmente preposições, interjeições, verbos auxiliares. Neste projeto foram executados experimentos com e sem stopwords para comparar a diferença de ganho em cada caso.

Implementação

Para comparar com o método base foi implementado um hiper-modelo que combina as estimativas dos classificadores através de um comitê e gera uma estimativa final. Semelhante a outros métodos de conjunto (*ensemble*), cada classificador é um membro de um comitê de classificadores e cada estimativa de um membro do comitê é chamado de voto. Assim, o voto da maioria será o voto do comitê. Além disso, a cada membro do comitê é associado um peso e, quando maior for o peso de um membro maior importância terá seu voto. O que diferenciará o método proposto dos outros métodos de ensemple existentes é como o peso de cada membro é calculado. Na figura abaixo, é mostrado um exemplo de comitê formado por 5 membros:

Comitê

Membros

Pares	M1	M2	M3	M4	M5	C
P1	V	F	F	F	F	F
P2	V	V	V	V	V	V
P3	V	V	F	F	V	V

Nesta figura tem-se um comitê formado por 5 membro $M1, M2, M3, M4, M5$ que classificam os pares $P1, P2, P3$. Os membros podem ser quaisquer classificadores, como os já descritos aqui, e os pares são os pares de questões do Quora. Para fins descritivos os valores estimados pelos classificadores, para o caso de deduplicação, são binários, 0 ou 1, neste caso representados por V (verdadeiro) e F (falso). Assim, $P1$ foi classificado como F pelo comitê, pois dos cinco membros quatro o estimaram como F, ou seja, o comitê estimou em 80% a probabilidade de $P1$ ser falso. Da mesma forma, $P2$ e $P3$ foram estimados como verdadeiro com as probabilidades de 100% e 60%, respectivamente.

O Calculo dos Pesos do Comitê e a Importância dos Pares

A ideia por trás do cálculo do peso dos membros do comitê vem da teoria do relacionamento de reforço mutuo [27]. Membros do comitê que acertam o rótulo de um determinado par recebem o reforço do peso desse par e membros que concordam com a estimativa do comitê devem ter seu peso reforçado ainda mais. Assim, deve-se observar as seguintes propriedades do algoritmo do cálculo de peso dos membros:

- **Peso do par**: o peso do par indica o nível de ganho de informação que pode oferecer ao aprendizado do classificador. Quanto maior mais difícil de ser estimado e quanto menor, mais fácil. Um membro que classifique corretamente um par com peso alto deve ser considerado melhor que os membros que acertam pares com pesos pequenos. Em resumo, o peso do par foi calculado da seguinte maneira:

$$Peso(V_p) = max(max(V_p) - min(V_p), 0.01)$$

Onde V_p é o vetor de similaridade do par p gerado a partir da aplicação das métricas de similaridade, $max(V_p)$ é o maior valor do vetor e, $min(V_p)$ é o menor valor do vetor. Lembrando que as similaridades estão normalizadas entre [0,1]. Pode-se observar que se os valores de similaridade do vetor forem parecidos o peso do par será baixo, ou seja, houve homogeneidade no cálculo da similaridade. Já no caso de haver muita diferença entre os valores de similaridade calculados, o peso será alto, mostrando que o par é difícil de ser estimado, pois houve grande divergência entre as métricas de similaridade.

- **Peso do membro**: o peso de um membro do comitê é calculado conforme algoritmo abaixo:

```
Para cada membro M do Comite C:
  if M.label == Par.label:
      if C.label == Par.label:
          M.weight += Par.weight * C.number_hits(Par)
       else:
          M.weight += Par.weight * C.number_erros(Par)
    else:
       if C.label == Par.label:
          M.weight += Par.weight / C.number_hits(Par)
       else:
          M.weight += Par.weight / C.number_erros(Par)
```

Assim, se um membro do comitê acertar o rótulo do par e concordar com o rótulo do comitê, ele receberá o valor do peso desse par multiplicado pela quantidade de membros que também acertaram junto com ele. No caso em que um membro acertar o rótulo do par, mas não concordar com o comitê, ou seja, o membro divergiu do comitê, esse membro receberá o peso do par multiplicado pelo número de membros que erraram o rótulo desse par. Quando o membro não acertar o rótulo do par, mas concorda com o comitê, esse membro receberá o valor do peso desse par dividido pelo número de membros que acertaram o rótulo desse par, neste caso, a minoria. E o pior caso, quando o membro não acertar o rótulo do par e nem concordar com o comitê, esse membro receberá o peso do par dividido pelo número de membros que também erraram, que neste caso é a maioria. Ao final, tem-se os membros do comitê ranqueados pelos seus respectivos pesos.

- **Peso do comitê**: o peso do comitê será a média do peso dos seus membros.

O comitê segue as mesmas etapas de um classificador. Primeiro precisa ser treinado (método *fit*) sobre o dataset de treino e depois testado sobre o dataset de teste. Assim é possível estimar a qualidade dos resultados do comitê. Logo, foram implementados os seguintes métodos ao comitê:

- *fit*: método do comitê que realizada o treinamento sobre o dataset de treino;
- *preditc*: método do comitê que estima o rótulo do conjunto de dados repassado;
- *predict_proba*: método do comitê que calcula a probabilidade dos pares do conjunto de treino serem tanto verdadeiros quanto falsos;

Logo, é possível comparar o método do comitê com qualquer um classificador descrito anteriormente.

Refinamento

Foi realizada uma melhoria no cálculo que estima o rótulo do comitê. Essa modificação se deu conforme descrito abaixo.

No modelo original um par era estimado como verdadeiro se a maioria dos membros estimassem como verdadeiro e vice-e-versa, gerando a estimativa final do comitê. O peso dos membros do comitê é calculado conforme essas estimativas. A partir daí, foi criado um ajuste nas estimativas dos membros do comitê, agora já se baseando nos próprios pesos do comitê. Considere-se a tabela abaixo que resume um processo de votação sobre alguns pares.

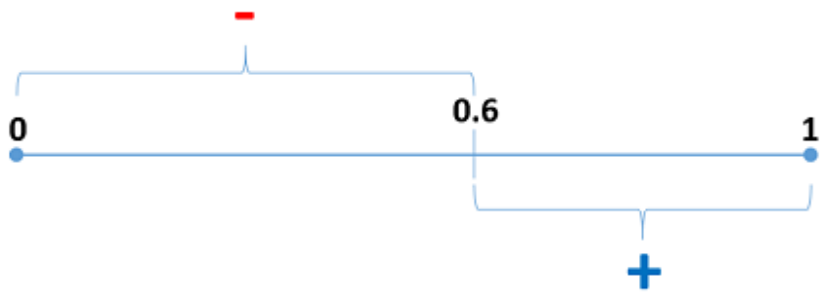
	M1	M2	M3	C	y	P_v	P_f
P1	1	1	1	1	1	1	0
P2	1	0	1	1	1	0,6	0,4
P3	0	0	1	0	1	0,3	0,7
P4	0	1	1	1	0	0,6	0,4
P5	0	0	0	0	0	0	1
W	0,75	0,375	0,735	0,62			

Nesta tabela os valores P1, P2, P3, P4 e P5 são os pares, W é o peso dos membros M1, M2 e M3 do comitê C. Os valores das células da tabela indicam o voto de cada membro em um par. Além disso, y é o rótulo real do par, P_v é a probabilidade dada pelo comitê do par Pi ser verdadeiro e Pv, de ser falso. O processo de ajuste das votações transforma os votos dos membros numa probabilidade proporcional a seus pesos, conforme cálculo abaixo:

$$P_1 = (1 - Limiar) - (Peso(M_i) * Limiar)$$

$$P_0 = Limiar - (Peso(M_i) * Limiar)$$

Onde P_1 é o cálculo da probabilidade quando o voto do membro M_i for 1 (verdadeiro) e P_0 para quando o voto do membro for 0 (falso), onde $Peso(M_i)$ e o peso do membro M_i, $Limiar$ é um limiar de corte, ilustrado na imagem abaixo.



Nesta imagem, tem-se uma linha que representa o domínio da probabilidade estimada por um membro do comitê, 0 é limite inferior e 1 o limite superior. 0.6 é limiar de corte, ou seja, qualquer valor de probabilidade estimada tal que $P \geq Limiar$, o par será considerado *verdadeiro* (+), caso contrário, *falso* (−). O valor do $Limiar$ é calculado a partir da média dos pesos dos membros do comitê.

Aplicando o ajuste nos valores da tabela acima, considerando o $Limiar = 0.62$, tem-se:

	M1	M2	M3	C_v	C_f	C	y
P1	0.845	0.6125	0.8357	0.7644	0.2356	1	1
P2	0.845	0.3875	0.8357	0.6894	0.3106	1	1
P3	0.155	0.3875	0.8357	0.4594	0.5406	0	1
P4	0.155	0.6125	0.8357	0.5344	0.4656	0	0
P5	0.155	0.3875	0.1643	0.2356	0.7644	0	0

Comparando o voto final do comitê C da primeira tabela com o voto após o ajuste, pode-se verificar que houve uma melhora na acurácia, pois no primeiro caso comitê cometeu dois erros e após o ajuste, só houve 1 erro. Uma análise mais completa desse ajuste será mostrada na seção Resultados.

Neste projeto, houve outra melhoria, mas relacionada as estruturas de dados utilizadas para realizar o processamento do método proposto. Foi criada uma estrutura de dados, chamada Infinity Array, para otimizar o uso de memória ao processar o conjunto de pares do *train* dataset, que é grande e ocupa bastante memória. Essa estrutura é uma lista na qual mantem somente um cache de seus elementos na memória e as outras partes não utilizadas ficam armazenadas em disco. Funciona de forma similar as estruturas de batch ou como cursores de banco de dados, mas sua utilização é transparente, como se fosse uma lista qualquer. Outra vantagem que os dados dessa lista podem ser salvos em disco para serem utilizados posteriormente.

IV. Resultados

Avaliação e Validaçao do Modelo

Foi realizado um conjunto de experimentos para avaliar e validar o modelo. Os experimentos foram configurados conforme a seguir:

1. Dataset: quora. Nº registros: 404mil. Tamanho treino: 60%. Tamanho teste: 40%.
2. Métricas de similaridade de texto: *Jaccard, JaccardSoundex, SoftTFIDF (com JaroWinkler, Soundex, Levenstein, MongeElkan, SmithWalterman), BagSim, Coseno, TFIDF, FullSim (com JaroWinkler, Soundex, Levenstein, MongeElkan, SmithWalterman)*.
3. Classificadores: *MultinomialNB, GaussianNB, SGDClassifier, MLPClassifier, DecisionTree, RandomForest, GradientBoosting*.
4. Método Proposto: *Comitê de Classificadores*
5. Métricas de avaliação: *F1, Logloss, Accuracy, Average Precision, Precision, Recall, Confusion Matrix, Precision-Recall Curve*.

Cada experimento avaliou a qualidade das predições dos classificadores em comparação à qualidade das predições do método proposto (comitê). Foram definidas 3 rodadas de experimento, onde cada rodada foi executada 3 vezes (com uma específica semente de aleatoriedade).

- **1ª Rodada**: com remoção de *stopwords*.
- **2ª Rodada**: com remoção de mínimo de *stopwords* (utiliza uma pequena lista com somente alguns stopwords).
- **3ª Rodada**: sem remoção de *stopwords*.

Uma rodada é formada das seguintes etapas:

1. Carregamento do dataset.
2. Pré-processamento dos dados
3. Blocagem e *tokenização*
4. Processamento dos pares e geração das features
5. Divisão aleatória das features entre treino e teste
6. Normalização das features
7. Treinamento dos classificadores
8. Formação e treinamento do comitê
9. Avaliação dos classificadores
10. Avaliação e ajuste do comitê
11. Geração dos resultados

Todos os detalhes de parâmetros dos classificadores estão no código fonte do projeto. Abaixo são apresentados os resultados obtidos em cada rodada.

As três primeiras tabelas de cada rodada mostram os valores obtidos pelas métricas Acuracy, Average Precision, Log Loss, Precision, Recall e F1. Cada tabela é o resultado de uma execução com uma semente distinta (0, 20 e 45).

As próximas tabelas seguintes mostram a média dos valores da Matriz de Confusão das três execuções.

Resultados da 1ª Rodada: com remoção de STOPWORDS

Semente 0	**accuracy**	**average_precision**	**f1**	**log_loss**	**precision**	**recall**
COMITE	0,66	0,50	0,66	0,54	0,52	0,89
DecisionTree	0,70	0,51	0,61	2,50	0,60	0,62
GaussianNB	0,68	0,50	0,64	2,58	0,54	0,77
GradientBoosting	0,70	0,50	0,60	0,72	0,59	0,62
MLPClassifier	0,70	0,50	0,58	0,52	0,59	0,57
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,63	0,52	0,62	0,63
SGD	0,68	0,47	0,52	0,55	0,58	0,48

Semente 20	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,66	0,50	0,66	0,54	0,52	0,89
DecisionTree	0,70	0,51	0,61	2,53	0,59	0,64
GaussianNB	0,67	0,50	0,63	2,61	0,54	0,77
GradientBoosting	0,70	0,50	0,60	0,69	0,58	0,63
MLP	0,70	0,50	0,60	0,52	0,58	0,62
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,63	0,52	0,62	0,63
SGD	0,68	0,48	0,55	0,55	0,57	0,53

Semente 45	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,66	0,50	0,66	0,54	0,52	0,89
DecisionTree	0,70	0,51	0,61	2,65	0,59	0,64
GaussianNB	0,68	0,50	0,64	2,59	0,54	0,77
GradientBoosting	0,70	0,50	0,60	0,70	0,59	0,62
MLP	0,71	0,51	0,60	0,52	0,60	0,60
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,63	0,53	0,62	0,64
SGD	0,68	0,47	0,54	0,55	0,58	0,50

Comitê	0	1	DTree	0	1	GaussianNB	0	1	RForest	0	1
0	0,62	0,38	**0**	0,77	0,23	**0**	0,77	0,23	**0**	0,52	0,48
1	0,23	0,77	**1**	0,43	0,57	**1**	0,37	0,63	**1**	0,11	0,89

GBoosting	0	1	MPL	0	1	MultiNB	0	1	SGD	0	1
0	0,75	0,25	**0**	0,80	0,20	**0**	1,00	0,00	**0**	0,74	0,26
1	0,38	0,62	**1**	0,52	0,48	**1**	1,00	0,00	**1**	0,38	0,62

Resultados da 2ª Rodada: com remoção mínima de STOPWORD

Semente 0	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,66	0,51	0,65	0,54	0,53	0,86
DecisionTree	0,70	0,51	0,61	2,89	0,59	0,62
GaussianNB	0,68	0,51	0,64	2,38	0,55	0,76
GradientBoosting	0,69	0,49	0,58	0,77	0,58	0,58
MLP	0,71	0,52	0,62	0,52	0,60	0,63
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,63	0,51	0,62	0,65
SGD	0,68	0,47	0,53	0,55	0,58	0,50

Semente 20	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,66	0,50	0,65	0,54	0,53	0,85
DecisionTree	0,70	0,51	0,61	2,96	0,59	0,63
GaussianNB	0,68	0,50	0,64	2,43	0,55	0,76
GradientBoosting	0,69	0,49	0,58	0,77	0,58	0,59
MLPClassifier	0,71	0,51	0,61	0,52	0,60	0,63
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,63	0,52	0,62	0,65
SGD	0,68	0,47	0,53	0,55	0,57	0,49

Semente 45	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,68	0,51	0,66	0,54	0,54	0,84
DecisionTree	0,70	0,50	0,60	3,01	0,59	0,60
GaussianNB	0,68	0,50	0,64	2,40	0,55	0,76
GradientBoosting	0,69	0,49	0,58	0,78	0,58	0,58
MLP	0,71	0,52	0,61	0,52	0,60	0,62
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,63	0,52	0,62	0,64
SGD	0,68	0,47	0,53	0,55	0,58	0,49

Comitê	0	1	DTree	0	1	GaussianNB	0	1	RForest	0	1
0	0,63	0,37	**0**	0,76	0,24	**0**	0,77	0,23	**0**	0,55	0,45
1	0,24	0,76	**1**	0,37	0,63	**1**	0,35	0,65	**1**	0,14	0,86

GBoosting	0	1	MPL	0	1	MultiNB	0	1	SGD	0	1
0	0,75	0,25	**0**	0,78	0,22	**0**	1,00	0,00	**0**	0,75	0,25
1	0,38	0,62	**1**	0,50	0,50	**1**	1,00	0,00	**1**	0,42	0,58

Resultados da 3ª Rodada: sem remoção de STOPWORD

Semente 0	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,69	0,52	0,66	0,54	0,56	0,79
DecisionTree	0,70	0,51	0,62	3,02	0,59	0,65
GaussianNB	0,68	0,51	0,64	2,28	0,55	0,76
GradientBoosting	0,69	0,49	0,57	0,80	0,57	0,57
MLP	0,71	0,52	0,61	0,51	0,61	0,62
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,64	0,51	0,62	0,65
SGD	0,68	0,47	0,52	0,55	0,58	0,48

Semente 20	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,69	0,52	0,65	0,54	0,56	0,80
DecisionTree	0,70	0,51	0,62	3,20	0,58	0,66
GaussianNB	0,68	0,50	0,63	2,32	0,55	0,76
GradientBoosting	0,69	0,49	0,58	0,80	0,58	0,58
MLPClassifier	0,71	0,52	0,62	0,51	0,61	0,63
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,64	0,52	0,62	0,66
SGD	0,68	0,47	0,52	0,55	0,57	0,48

Semente 45	accuracy	average_precision	f1	log_loss	precision	recall
COMITE	0,69	0,52	0,66	0,54	0,56	0,80
DecisionTree	0,70	0,51	0,61	3,41	0,58	0,64
GaussianNB	0,68	0,50	0,64	2,29	0,55	0,76
GradientBoosting	0,68	0,48	0,57	0,82	0,57	0,57
MLPClassifier	0,72	0,52	0,62	0,51	0,61	0,63
MultinomialNB	0,63	0,37	0,00	0,64	0,00	0,00
RandomForest	0,72	0,53	0,64	0,52	0,62	0,66
SGD	0,68	0,47	0,52	0,55	0,58	0,48

Comitê	0	1	DTree	0	1	GaussianNB	0	1	RForest	0	1
0	0,64	0,36	**0**	0,77	0,23	**0**	0,77	0,23	**0**	0,64	0,36
1	0,24	0,76	**1**	0,38	0,62	**1**	0,35	0,65	**1**	0,21	0,79

GBoosting	0	1	MPL	0	1	MultiNB	0	1	SGD	0	1
0	0,73	0,27	**0**	0,79	0,21	**0**	1,00	0,00	**0**	0,75	0,25
1	0,35	0,65	**1**	0,52	0,48	**1**	1,00	0,00	**1**	0,43	0,57

Justificação

Os resultados mostraram, de forma geral, que os classificadores obtiveram resultados bem parecidos. A variação dos valores médios de acuracy foi de 6%. Houve pouca diferença entre as três execuções por rodada. Também não houve muita diferença entre os valores obtidos entre as rodadas, mas, o método proposto obteve os melhores resultados com remoção de todas as stopword. Já entre os outros classificadores o ganho foi pouco. Mesmo assim, o processo de limpeza do *train* dataset com a remoção de STOPWORDS, transformação e normalização trouxe ganhos.

O classificador com melhor Acuracy foi o RandomForest, alcançando 72%. Todos classificadores obtiveram baixo Average Precision, em média 50%, o maior, 51%, foi alcançado pelo classificador DecisionTree. Como a métrica Log Loss reflete, na maioria dos casos, a métrica Acuracy, o melhor Log Loss foi obtido também pelo classificador RandomForest, com o valor de 0,52. Entretanto, o que avalia melhor a qualidade dos classificadores num processo de identificação de registros duplicados são as métricas Precision, Recall e F1. Sendo que, como F1 é a média harmônica entre Precision e Recall, os melhores valore de F1 refletem o melhor resultado do classificador. Neste caso o método proposto, COMITÊ, alcançou o melhor F1, no valor de 66%. Apesar de maior valor de Precicion ter sido alcançado pelo classificador RandomForest, no valor de 62%. O melhor valor de Recall também foi obtido pelo COMITÊ, que foi de 89%, no melhor caso.

Os resultados dados pela Matriz de Confusão refletem os valores obtidos de Precision, Recall e F1. Mas nesse formato é possível ter uma ideia melhor de como os classificadores se comportaram em relação aos pares que foram classificados como Falso ou como Verdadeiro. Lembrando que, neste caso, os valores contidos na diagonal de cada matriz devem ser os mais próximos de 1, que é quando o acerto é de 100%.

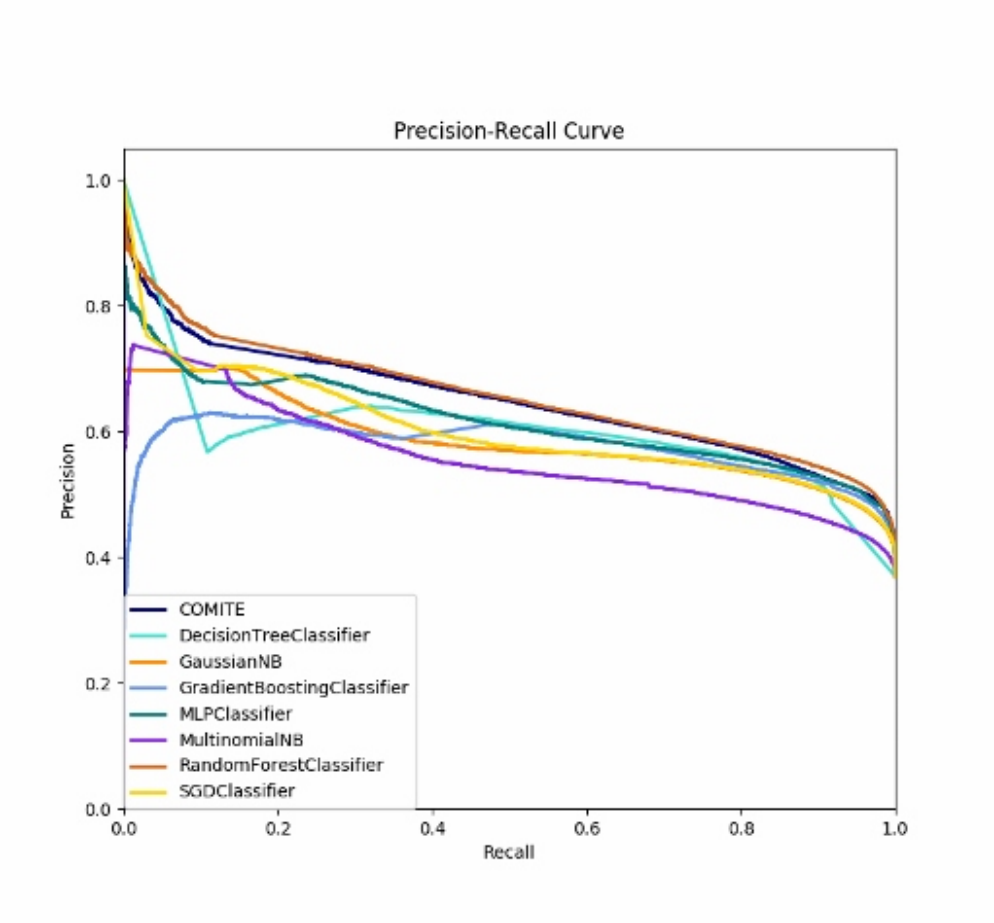
De forma geral, foi possíve verificar que o método proposto (Comitê) obteve resultados razoavelmente melhores que o método base (Random Forest). Ou, pode-se interpretar que os resultados alcançados pelo Comitê foram de responsabilidade de seu membro mais importante, o Random Forest.

V. Conclusão

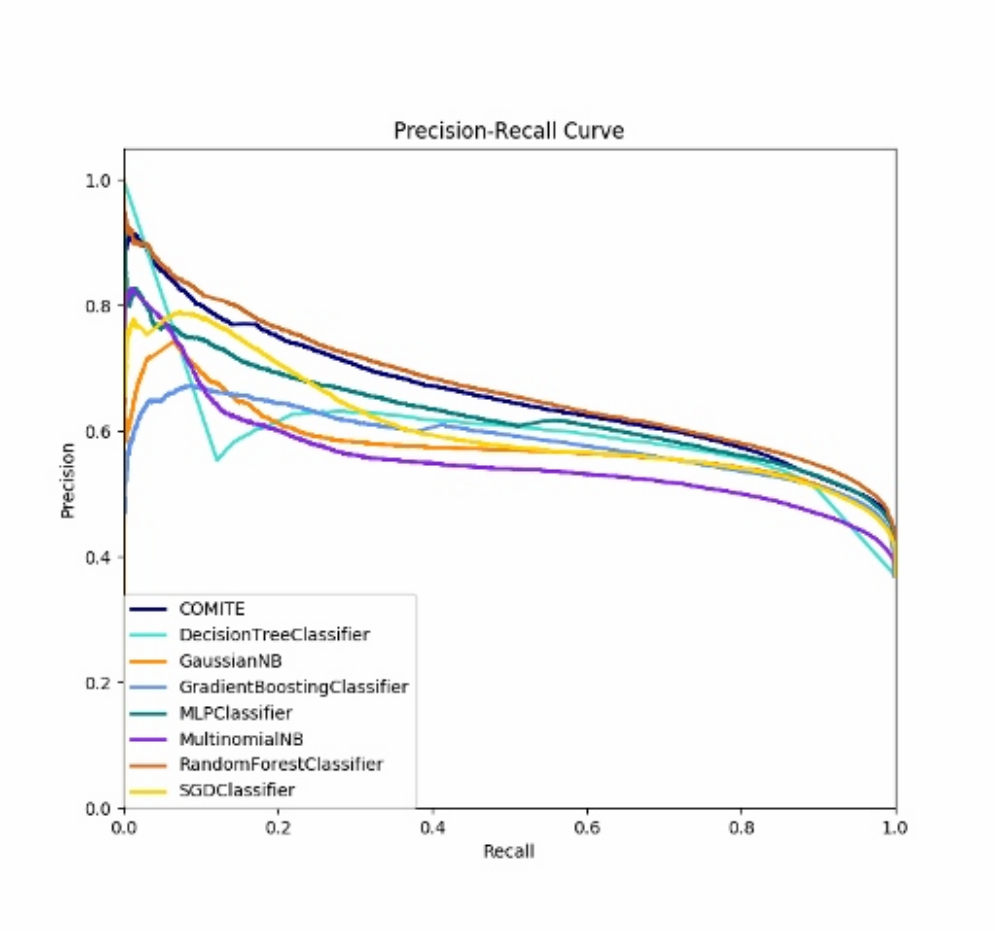
Visualizaçao de Forma Livre

Aqui são mostradas as curvas de Precisão X Revocção (Precision-Recall Curves). Com esses gráficos das curvas é possível ver de forma mais clara como os classificadores se comportaram um em relação ao outro, é possível ver o que obteve melhor resultado e pior também.

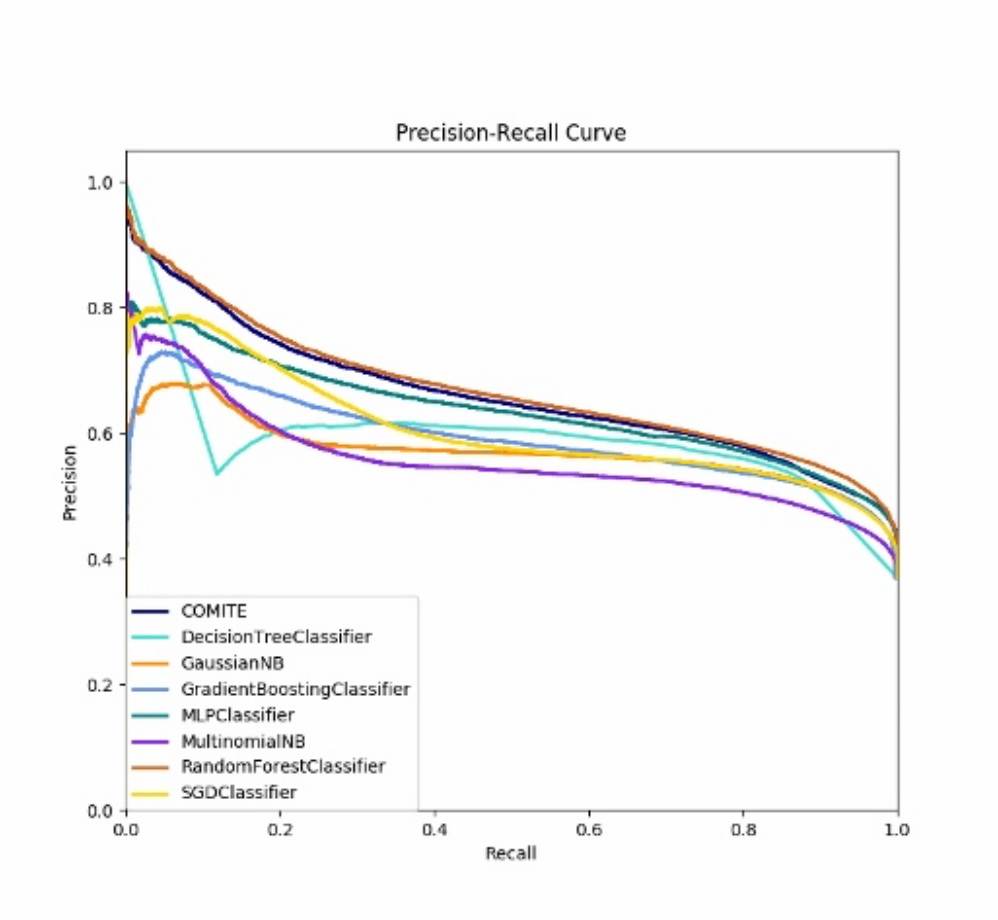
1ª Rodada



2ª Rodada


Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
COMITE
DecisionTreeClassifier
GaussianNB
GradientBoostingClassifier
MLPClassifier
MultinomialNB
RandomForestClassifier
SGDClassifier

3ª Rodada



Como pode ser observado, as curvas, em ambas rodadas, mostram que o método proposto (COMITÊ) obteve, na média, qualidade semelhante ao do classificador RandomForest na tarefa de classificar os pares do Cora. Seguido pelo classificador MLP (Rede Neural). Por outro lado, o pior resultado foi do classificador MultinomialNB.

Reflexão

O processo de identificar registro duplicados continua sendo um grande desafio, principalmente quando se trata de se identificar pares de registros no qual o valor semântico tem peso maior do que o valor sintático, como é o caso do Dataset do Quora. Por isso da própria plataforma Quora resolver patrocinar uma competição no Kaggle. Neste projeto resolveu-se utilizar das técnicas tradicionais de Data Mining para tentar classificar os pares de perguntas do Quora. Entretanto, conforme se pôde observar nos resultados a utilização de técnicas de similaridade, com foco na sintaxe dos dados, não obtiveram resultados altos, porém foram expressivos. Isso mostra um novo caminho a se investir na resolução desse tipo de problema: usar técnicas de análise semântica de texto. Como por exemplo, existem os métodos LSI[28] e WORD2VEC[29] que poderiam ser usados. Este último utiliza Deep Learning.

Os resultados obtidos pelo método proposto, o comitê de classificadores, conseguiu superar os resultados individuais de seus membros. Isso mostra que a combinação de métodos tradicionais de Aprendizado de Máquina traz ganho real aos resultados. Faz-se necessário tester esse método em outras bases e fazer outros experimentos variando o tamanho do comitê, utilizar outros classificadores como membros e usar outras *features* para estimar corretamente pares de registros na tarefa de deduplicação.

Além disso, este trabalho mostrou e implementou todas as etapas de um processo de limpeza de dados, como *tokenização*, blocagem, uso de métricas de similaridade de texto, técnicas de indexação e recuperação de documentos, que poderão servir como base para outras tarefas de Data Mining que possam utilizar Aprendizado de Máquina e Inteligência Artificial.

Melhorias

Como melhorias e trabalhos futuros foram sugeridos:

- Uso de métricas semânticas para gerar as *features*;
- Usar GPU para realizar o cálculo da similaridade dos pares de questões, pois esse cálculo é computacionalmente caro;
- Testar o método proposto em outros datasets;
- Utilizar Aprendizado Ativo para auxiliar o comitê quando, ao classificar um par, tiver dúvida se é verdadeiro ou falso.

VI. Referências

1. https://www.kaggle.com/c/quora-question-pairs↩

2. https://pt.wikipedia.org/wiki/Processamento_de_linguagem_natural↩

3. https://en.wikipedia.org/wiki/Latent_semantic_analysis↩

4. www.cs.cmu.edu/~wcohen/postscript/ijcai-ws-2003.pdf↩

5. http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#sphx-glr-auto-examples-model-selection-plot-precision-recall-py↩

6. https://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/↩

7. https://en.wikipedia.org/wiki/Loss_functions_for_classification↩

8. https://en.wikipedia.org/wiki/Stop_words↩

9. BORUCH, Robert; STROMSDORFER, Ernst. Exact matching of micro data sets in social research: Benefits and problems. In: Proceedings of the Workshop on Exact Matching Methodologies. [S.l.: s.n.], 1985. p. 145–153.↩

10. HOWE, G. R.; LINDSAY, J. A generalized iterative record linkage computer system for use in medical follow-up studies. Computers and Biomedical Research, v. 14, p. 327–340, 1981.↩↩

11. de FREITAS, Júnio da Silva; PAPPA, G. L. ; GONÇALVES, Marcos A. ; VELOSO, A ; MOURA, Edleno Silva de ; SILVA, Altigran Soares da. Active Learning Genetic Programming for Record Deduplication. In: IEEE Congress on Evolutionary Computation (CEC), 2010, Barcelona. IEEE Congress on Evolutionary Computation (CEC), 2010.↩

12. JARO, Matthew A. UNIMATCH: A Record Linkage System. User's Manual. [S.l.], 1976. Disponível em: http://www.bibsonomy.org/bibtex/242f41fb75a77fec25383ce0d2120e928-/pirot.↩

13. WINKLER, William E. The State of Record Linkage and Current Research Problems. Washington, DC, 1999.↩

14. BILENKO, M.; MOONEY, R.; COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. Adaptive name matching in information integration. Intelligent Systems, IEEE, v. 18, n. 5, p. 16–23, Sep/Oct 2003. ISSN 1541-1672↩↩

15. COHEN, William W.; RAVIKUMAR, Pradeep; FIENBERG, Stephen E. A comparison of string distance metrics for name-matching tasks. In: IJCAI-2003. Proceedings of the Workshop on Information Integration on the Web. Acapulco, Mexico, 2003. p. 73–78.↩

16. BOURNE, Charles P.; FORD, Donald F. A study of methods for systematically abbreviating english words and names. J. ACM, ACM, New York, NY, USA, v. 8, n. 4, p. 538–552, 1961. ISSN 0004-5411.↩

17. Monge, A., and Elkan, C. 1996. The field-matching problem: algorithm and applications. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.↩

18. https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm↩↩

19. http://scikit-learn.org/stable/modules/tree.html↩

20. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html↩

21. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html↩

22. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html↩

23. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html↩

24. http://scikit-learn.org/stable/modules/ensemble.html#forest↩

25. http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting↩

26. http://ieeexplore.ieee.org/document/6263211/↩

27. https://en.wikipedia.org/wiki/HITS_algorithm↩

28. https://en.wikipedia.org/wiki/Latent_semantic_analysis↩

29. https://en.wikipedia.org/wiki/Word2vec↩