TÉCNICO LISBOA

Detecção de Paráfrases e Aplicações na Pesquisa de Respostas em Bases de Dados de Perguntas Frequentes

António Pedro Pires Amaral

Dissertação para obtenção do grau Mestre em

Engenharia Informática e de Computadores

Júri

Presidente: Prof. Doutor Pedro Manuel Moreira Vaz Antunes de Sousa
Orientador: Prof. Doutor Bruno Emanuel da Graça Martins
Co-orientador: Prof. Doutor Pável Pereira Calado
Vogal: Prof.ª Doutora Maria Luísa Torres Ribeiro Marques da Silva Coheur

Novembro 2013

Abstract

This work revisits the task of identifying paraphrases. We propose to address the task through supervised machine learning, training classification models based on ensembles of trees, which use features that mostly correspond to string similarity metrics relying on lexical information. The most innovative contributions of this work relate to (i) the usage of classification methods based on tree ensembles, (ii) the combination of machine translation metrics with other types of features, and (iii) the usage of similarity features based on distributional word clustering. We report on a set of experiments that used the well-known Microsoft Research Paraphrase Corpus, in which we achieved a classification accuracy of 0.77, and an F1 measure of 0.84. We therefore show that out-of-the-box learning algorithms and relatively simple features can obtain state-of-the-art results in this task. We also applied the proposed method for paraphrase detection to a task of querying a database of frequently asked questions (FAQ Retrieval). Using the data from an international contest FireFAQ, we achieved a precision at the first position of the ranked list of results of 0.83 and 0.73, respectively with the data from FireFAQ 2011 and FireFAQ 2012. Thus, we have empirically proven that the same algorithms can be used either in the detection of paraphrase relations between sentences, or between one query and a set of questions in a FAQ database.

Keywords: Paraphrase Detection , FAQ Retrieval , String Similarity Measures , Supervised Classification , Machine Learning

Sumário

Este trabalho trata da tarefa de identificação de paráfrases. Propomos tratar desta tarefa usando aprendizagem automática supervisionada, treinado modelos de classificação baseados em árvores de decisão que usam características correspondentes a medidas de similaridade entre strings, baseadas essencialmente em informação lexical. As contribuições mais inovadoras deste trabalho relacionam-se com (i) o uso de métodos de classificação do atual estado da arte baseados em árvores de decisão, (ii) a combinação de medidas da área da tradução automática com outras características, e (iii) o uso de características baseadas em medidas de similaridade entre strings, as quais utilizam informação obtida com técnicas de *word clustering*. É reportado neste artigo um conjunto de experiências que usam o conhecido Microsoft Research Paraphrase Corpus, em que conseguimos uma exatidão de 0.77 e uma medida F1 de 0.84. Assim, é demonstrado que algoritmos de aprendizagem automática usando características relativamente simples podem obter resultados semelhantes aos resultados do actual estado-da-arte nesta tarefa. Aplicámos também o método proposto para a deteção de paráfrases a um problema de pesquisa em bases de dados de perguntas frequentes (FAQ Retrieval). Usando os dados do concurso internacional FireFAQ, obtivemos uma precisão na primeira posição da lista de resultados de 0.83 e 0.73, respetivamente com os dados das edições FireFAQ 2011 e FireFAQ 2012. Desta forma, prova-se empiricamente que os mesmos algoritmos podem ser usados quer na detecção de relações de paráfrase entre duas frases, quer na detecção de uma paráfrase para com um conjunto de perguntas frequentes (FAQ).

Keywords: Detecção de Paráfrases , Recuperação de Informação em FAQs , Métricas de Similaridade entre Strings , Classificação Supervisionada , Aprendizagem Automática

Agradecimentos

Começo por agradecer e dedicar esta tese ao meu avô que foi sempre uma pessoa que respeitei e admirei durante toda a minha vida e que hoje, infelizmente, não pode celebrar o fim deste capítulo da minha vida.

Um agradecimento especial é devido ao Professor Bruno Martins que sempre mostrou uma dedicação e empenho na orientação desta tese. Ainda hoje, não me esqueço do dia em que encontrei o Professor Bruno Martins no metro quando estava com problemas na produção desta tese ao mesmo tempo que trabalhava, tendo demonstrado que estava sempre disponível para corrigir, orientar e ter novas ideias para melhorar este trabalho.

Gostaria também de agradecer à minha família, que me deu sempre um apoio incondicional à realização da tese e de tudo o que fiz na minha vida.

Um agradecimento particular também é devido a uma amiga que me incentivou a voltar à tese enquanto estive a trabalhar, a Maria João Pereira.

Agradeço ainda a uma antiga colega de trabalho, Diana Salavisa, que me incentivou a concluir o mestrado enquanto estive a trabalhar.

Por último, um agradecimento ao amigo que trabalhou mais tempo comigo, o Hugo Guimarães. Agradeço ao Hugo por todos os debates de ideias e horas de trabalho.

Lisboa, Outubro 2013

António Amaral

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O acto de parafrasear é geralmente definido como a reformatação (ou o re-uso) de texto, dando o mesmo significado de uma outra forma. O problema de modelar as relações entre paráfrases em expressões de língua natural tem recentemente atraído muito interesse. Para linguístas computacionais, resolver este problema pode mostrar como modelar melhor a semântica de uma frase. Para engenheiros de linguagem natural, esta tarefa tem importantes aplicações em diferentes tipos de tarefas práticas envolvendo medidas de sobreposição semântica entre frases, como por exemplo no contexto de sumarizadores automáticos abstrativos, sistemas de resposta a perguntas, sistemas de recuperação de informação em FAQs, sistemas de tradução automática, ou sistemas automáticos de verificação de direitos de autor.

Em suma, o problema apresentado relaciona-se com a identificação de paráfrases, o que consiste em saber se duas frases têm o mesmo significado. Embora a identificação de paráfrases seja definida em termos semânticos, é muitas vezes resolvida por classificadores estatísticos fazendo uso de características de sobreposição lexical superficial, e de sobreposição de n-gramas (e.g., métodos típicos contam com o uso de técnicas de correspondência lexical, em que a similaridade entre dois textos candidatos é processada como uma função do número de correspondências entre as sequências de tokens partilhados entre os textos). Recentemente, vários autores têm experimentado o uso de métodos de aprendizagem automática supervisionada para combinar múltiplos tipos de características, incluindo a similaridade entre palavras usando o WordNet ou usando métodos de *word clustering*, ou até funções sintáticas baseadas em árvores de dependência gramatical. No entanto, embora a introdução de características com informação linguística tenha mostrado um melhoramento na performance, estes métodos têm também as suas desvantagens em termos de performance computacional, e em termos dos recursos necessários e no processamento linguístico. Temos, por exemplo, que a maioria

do trabalho anteriormente feito na área focou-se na língua Inglesa, usando como referência o Microsoft Research Paraphrase Corpus. Contudo, aplicações noutras línguas não podem contar com recursos como o WordNet ou com bibliotecas robustas para o processamento da língua natural, para extrair informação sintática, dado que estes recursos podem não estão disponíveis.

Neste trabalho, é proposto e avaliado um método para a identificação automática de paráfrases, o qual usa algoritmos de classificação do actual estado da arte, e características relativamente simples que são independentes do domínio.

1.1 Hipótese e Metodologia

Neste artigo, revisitamos o problema de identificação de paráfrases, modelando o problema como uma tarefa de classificação que nós propomos resolver usando algoritmos baseados na combinação de múltiplas árvores de decisão, e usando ainda um vasto conjunto de características que essencialmente correspondem a métricas de similaridade entre strings, baseadas em informação lexical. As contribuições mais inovadoras deste trabalho relacionam-se com (i) o uso de métodos de classificação baseados em conjuntos de árvores, (ii) a combinação de medidas da área da tradução automática com outras características, e (iii) o uso de características baseadas em medidas de similaridade entre strings, as quais usam os resultados de algoritmos de *word clustering*.

Procurou-se demonstrar que algoritmos de aprendizagem automática usando características relativamente simples podem obter resultados semelhantes aos resultados do actual estado-da-arte na identificação de relações de paráfrases entre duas frases. Para tal, usámos os dados do Microsoft Reaseach Paraphrase Corpus. Usando os dados do concurso internacional FireFAQ, mostramos ainda que uma abordagem de deteção de paráfrases pode ser uma boa abordagem para encontrar uma pergunta com o mesmo significado que uma dada interrogação, numa base de dados de perguntas frequentes. Esta abordagem usa as mesmas características que usamos para o Microsoft Research Paraphrase Corpus.

1.2 Contribuições

É reportado neste trabalho um conjunto de experiências que usam o conhecido Microsoft Research Paraphrase Corpus, no qual conseguimos uma classificação com uma exatidão de 0.77 e uma medida F1 de 0.84. Assim, é mostrado empiricamente que algoritmos de aprendizagem

usando características relativamente simples podem obter resultados semelhantes aos resultados do actual estado-da-arte na tarefa de identificar paráfrases, a qual tem uma importância cada vez maior.

Foi também efetuado um conjunto de experiências com os dados dos concursos FireFAQ 2011 e FireFAQ 2012, nos quais obtivemos uma precisão na primeira posição da lista ordenada de resultados 0.83 e 0.73, respetivamente. Desta forma, mostrámos empiricamente que os algoritmos usados para o Microsoft Research Paraphrase Corpus podem ser usados também numa tarefa de pesquisa de bases de dados de FAQs.

1.3 Organização do Documento

O resto desta dissertação está organizado da seguinte forma: o Capítulo 2 apresenta conceitos fundamentais e trabalho relacionado na área. O Capítulo 3 apresenta o método proposto, detalhando, por exemplo, as características que foram consideradas. O Capítulo 4 apresenta os resultados da validação experimental. Por último, o Capítulo 5 apresenta as conclusões, e possíveis direções para trabalho futuro.

Capítulo 2

Conceitos e Trabalho Relacionado

Neste capítulo são apresentados os conceitos principais das áreas científicas onde o trabalho se insere, assim como os trabalhos relacionados mais relevantes.

2.1 Conceitos Fundamentais

O trabalho proposto relaciona-se com a forma como uma frase pode ser interpretada e classificada como tendo o mesmo significado que outra. A melhor forma de ter ambas as funcionalidades é fazendo uso de mecanismos que permitam inferir da similaridade entre *strings*. Assim, numa primeira parte, esta secção trata de descrever diversos tipos de medidas de similaridade entre *strings*, com exemplos de algoritmos usados em cada um desses tipos. Após a primeira parte, esta secção introduz ainda alguns conceitos fundamentais relacionados com tarefas de aprendizagem automática.

2.1.1 Métricas de Similaridade Entre *Strings*

As técnicas de medição de similaridade entre *strings* permitem-nos determinar a diferença entre duas *strings*, resultando num valor entre 0, se as *strings* são completamente diferentes, ou de 1 se as *strings* têm exatamente os mesmo caracteres e na mesma ordem.

Por exemplo no contexto da detecção de paráfrases, uma forma de verificar se duas *strings* significam o mesmo passa por usar um valor limiar sobre a similaridade entre elas, que dite o

limite de aceitação entre duas *strings* serem consideradas como significando o mesmo.

De seguida, são explicados alguns dos algoritmos que foram usados no sistema no passado para desenvolver sistemas que envolviam a identificação do grau de similaridade entre *strings*.

2.1.1.1 Métricas Baseadas em Caracteres

As métricas de similaridade baseadas em caracteres detetam as diferenças entre *strings* usando comparações entre os caracteres de cada *string*.

Esta abordagem é particularmente útil para considerar duas *strings* como semelhantes na presença de erros de digitação.

Um algoritmo particular é a distância de Levenshtein, também chamado de distância de edição, embora existam outros algoritmos de distância de edição. O algoritmo ficou com o nome do seu criador, Vladimir Levenshtein (Levenshtein, 1966). O objetivo do algoritmo de distância de Levenshtein é calcular o número de edições que têm de ocorrer para que uma *string* seja igual a outra. Estas edições podem ser a inserção de um caracter numa dada posição, a eliminação de um caracter numa dada posição, ou a substituição de um caracter por outro.

Este algoritmo usa uma matriz para calcular o valor final da distância de Levenshtein. Esta matriz tem um número N de linhas, em que $N-1$ é o número caracteres da primeira frase; e tem um número M de colunas, em que $M-1$ é o número caracteres da segunda frase.

A matriz é inicializada com valores base e depois é preenchida com valores computados sob os valores previamente calculados como é mostrado na Equação 2.1.

$$LD(i,j) = min \begin{cases} \text{caso caracteres coincidam:} & \\ LD(i-1,j-1) & \\ \text{caso não coincidam:} & \\ LD(i-1,j-1)+1 & \text{(substituir)} \\ LD(i-1,j)+1 & \text{(eliminar)} \\ LD(i,j-1)+1 & \text{(inserir)} \end{cases} \tag{2.1}$$

O processo eficiente de resolver problemas recursivos deste tipo, com valores computados de valores calculados anteriormente, é também conhecido como programação dinâmica.

Como se pode verificar na Figura 2.1, existem 3 passos essenciais. No primeiro faz-se uma inicialização. De seguida, no segundo passo, é feita uma iteração do algoritmo para cada posição. Esta segunda fase está representada no meio da figura, podendo-se verificar que

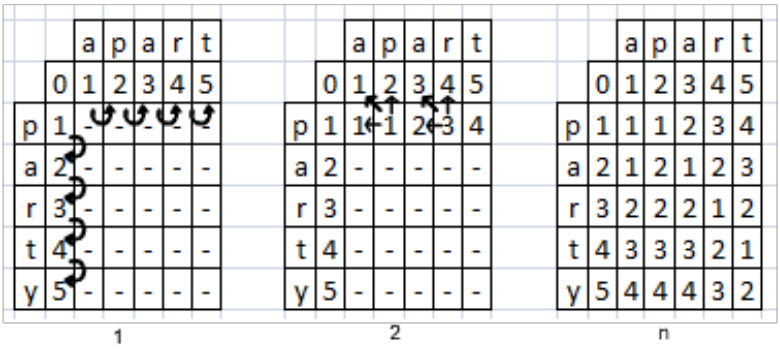

Figura 2.1: Diferentes passos do algoritmo da distância de Levenshtein.

para cada caracter da segunda *string* existe uma comparação para com a primeira *string*. É calculado o custo da eliminação (representado com a seta apontando para cima), da inserção (representado com uma seta apontando para a esquerda), e da substituição (representado com a seta apontado na diagonal), baseado nos valores das posições correspondentes. Por último, é representada na figura a matriz preenchida, onde o valor da última célula representa a distância de edição.

A similaridade entre duas *strings*, baseada na distância de Levenshtein, pode ser calculada por uma fórmula que normaliza o valor da distância de Levenshtein para um valor no intervalo [0,1].

$$sim(s_1, s_2) = 1 - \frac{d}{\max(|s_1|, |s_2|)} \tag{2.2}$$

A Fórmula 2.2 tem como parâmetros a distância calculada pelo algoritmo da distância de Levenshtein, e o número de caracteres da maior *string*, representado por $\max(|s_1|, |s_2|)$.

Neste projeto, a distância de Levenshtein é usada para detetar erros de digitação inseridos nas palavras da questão do utilizador.

Uma outra medida baseada em caracteres é a distância de Jaro-Winkler. A distância de Jaro-Winkler foi criada como sendo uma variante da distância de Jaro (Jaro, 1989; Malakasiotis, 2009). Esta métrica é muito usada, por exemplo, na deteção de nomes duplicados.

Em dois nomes como *William J.Smith* e *Smith, W.J.*, a distância de Levenshtein não ajudaria pois é uma abordagem caracter a caracter, sem ter em conta que conceitos similares podem ser representados de formas diferentes (e.g., com abreviaturas).

Usando o exemplo, a similaridade entre *William J. Smith* e *Smith, W.J.* usando a distância de

Levenshtein é de 0,1875. Alternativamente, a similaridade usando o algoritmo de Jaro-Winkler seria de 0,8377, aproximadamente. A distância de Jaro-Winkler é calculada em duas fases. Primeiro, a distância de Jaro entre duas *strings* s_1 e s_2 é calculada usando uma fórmula que inclui a possibilidade dos caracteres estarem fora de posição.

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \tag{2.3}$$

A Fórmula 2.3 tem como parâmetros o número de caracteres da primeira *string*, representado por $|s_1|$, o número de caracteres da segunda *string*, representado por $|s_2|$, e o número de caracteres correspondentes nas duas *strings*. Estes caracteres podem não estar na posição equivalente i, mas se estiverem dentro dos limites de uma janela de comparação, eles são aceites como sendo caracteres correspondentes. Estes limites são definidos segundo a Fórmula 2.4 e representados por m. Por último, o número de transposições é representado por t, sendo o número de caracteres correspondentes, mas com ordem de sequência diferente (e.g. ab e ba) dividido por 2.

$$\left[i - \left(\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1\right), i + \left(\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1\right)\right] \tag{2.4}$$

De seguida, a distância de Jaro-Winkler é determinada usando como um dos parâmetros o resultado da Fórmula 2.3.

$$d_w = d_j + (\ell p(1 - d_j)) \tag{2.5}$$

A Fórmula 2.5 tem como parâmetros o resultado da Fórmula 2.3, como já foi dito e representado por d_j, o número de caracteres comuns no início de uma *string*, até 4 caracteres e representado por ℓ, e um fator de escalabilidade p, cujo valor tem de ser menor do que 0,25.

$$d_w = d_j + (\ell p(1 - d_j)) = d_j + \ell p - \ell p d_j = (1 - \ell p)d_j + \ell p \tag{2.6}$$

Salienta-se a importância que é dada ao prefixo ser igual, em detrimento da distância de Jaro, como é provado na Equação 2.6. Se o valor de p for 0,25 e ℓ for 4, o valor de d_j é ignorado, mesmo que a palavra tenha mais do que 4 caracteres.

Este algoritmo deve ser usado em *strings* curtas, que na maioria das vezes consistam em apenas poucas palavras ou uma se possível, uma vez que o impacto do prefixo na primeira palavra é menor quanto mais longe a palavra estiver da primeira.

2.1.1.2 Métricas Baseadas em Fonemas

Esta abordagem foca-se em analisar os sons que constituem uma dada palavra ou *string* a comparar. Estes sons não são literalmente comparados, mas em vez disso é comparada uma codificação dos sons. Por exemplo para a palavra *Indian*, usando o algoritmo Soundex, é representada como I535. Uma palavra com o mesmo som mas escrita diferente, por exemplo *Indean* é também representada como, I535 no mesmo algoritmo.

O algoritmo Soundex recebe uma palavra e retorna a versão codificada da palavra, representando a forma como a palavra soa (Odell and Russell). O Soundex é a base de vários algoritmos baseados em fonemas, codificando somente consoantes, exceto a primeira letra.

Como primeiro passo, o algoritmo deixa a primeira letra da palavra como estava, independentemente de ser consoante ou vogal. Depois, elimina todas as vogais, 'h', 'w' e 'y' que apareçam. Por cada outra letra, codifica-se com um número entre 1 e 6. Por último, se dois números adjacentes forem iguais, elimina-se um deles.

O algoritmo Metaphone é um melhoramento feito sobre o Soundex, que codifica mais informação sobre sons específicos de consoantes na língua inglesa (Philips, 1990). Por exemplo, algumas consoantes são agrupadas com outras consoantes que não são ouvidas. Novamente, a maioria das vogais não são tidas em conta a menos que seja a primeira letra.

Existem ainda duas variantes mais modernas do Metaphone. No Double Metaphone, duas análises diferentes da palavra são feitas e comparadas, para que sejam encontrados outros tipos de erros. O Metaphone 3 permite ainda ter em conta o som das vogais.

2.1.1.3 Métricas Baseadas em N-Gramas ou Palavras

As medidas de similaridade entre *strings* baseadas em n-gramas ou palavras vêem as *strings* como grupos de *tokens*, e usam propriedades destes grupos para processar os valores de similaridade. Por outras palavras, a similaridade baseada em n-gramas ou palavras divide uma *string* em diferentes partes que seguem um padrão específico. Estes grupos podem ser desde pequenos grupos de 2 caracteres, até grupos constituídos por cada palavra.

Estas funções são úteis para encontrar *strings* quase duplicadas em casos onde as convenções de digitação mudam a forma como as palavras são escritas (e.g., "John Smith"e "Smith, John").

Por exemplo, a medida de Jaccard, inventada em 1901 (Jaccard, 1901), pode ser usada para n-gramas ou para outros tipos de conjuntos, correspondendo à seguinte equação:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.7}$$

A Fórmula 2.7 leva em conta os *tokens* em comum entre as duas *strings*, representado por $|A \cap B|$, e o número de palavras ou n-gramas diferentes que existem nas duas *strings*, representado por $|A \cup B|$.

Já o coeficiente de sobreposição (do Inglês *Overlap Coefficient*) corresponde à seguinte equação:

$$Overlap(A,B) = \frac{|A \cap B|}{min(|A|,|B|)} \tag{2.8}$$

A Fórmula 2.8 leva em conta os *tokens* em comum entre as duas *strings*, representado por $|A \cap B|$, e o menor número de palavras ou n-gramas entre A e B, representado por $min(|A|,|B|)$.

A medida de Dice (Dice, 1945) corresponde à seguinte equação:

$$s = \frac{2\,|X \bigcap Y|}{|X| + |Y|} \tag{2.9}$$

A Fórmula 2.9 leva em conta os *tokens* em comum entre as duas *strings*, representados por $|X \bigcap Y|$, e o tamanho de cada *string*, representados por $|X|$ e $|Y|$. O valor de 2 é um fator de escalabilidade para normalizar a fórmula.

Finalmente, a similaridade entre duas strings pode ser calculada com base no coseno do ângulo entre representações vectoriais para as suas palavras. Nestas representações, os pesos TF-IDF avaliam o quão importante uma palavra é num dado documento, de entre um grupo de documentos. Esta operação tem uma importante relevância neste trabalho sendo muito usada em *top-k retrieval* pela sua alta performance (Zhua et al., 2009).

O peso de TF-IDF consiste na multiplicação de dois valores mais simples, nomeadamente a frequência do termo (do inglês *term frequency*) e a frequência inversa do documento (do inglês *inverse document frequency*).

Inicialmente, a operação de frequência do termo retorna um valor proporcional ao número de vezes que a palavra aparece num dado documento.

Existem, no entanto, algumas variações do cálculo de TF. Por exemplo, para resolver o problema de um documento que tenha mais palavras poder ter um maior TF, o resultado pode ser simplesmente a divisão entre a frequência e o número de palavras no documento.

De seguida, a frequência inversa do documento é uma operação que retorna um valor que é menor quanto mais frequente uma palavra for num grupo de documentos. A maior motivação

por detrás do uso da operação de IDF é diminuir, ou mesmo eliminar, o peso das palavras muito frequentes que não sejam essenciais ou necessárias para a procura.

$$log(\frac{|N|}{|\{n : t \in N\}| + 1}) \tag{2.10}$$

A Fórmula 2.10 mostra a forma mais comum de calcular o IDF. Nesta fórmula, é representado o número de documentos por $|N|$, e o número de documentos em que o termo t existe por $|\{n : t \in N\}|$. De notar que o valor de 1 no denominador é um fator para evitar a divisão por 0 no caso do termo t não existir em nenhum documento.

Após a operação de TF-IDF estar completa, cada palavra diferente de cada documento terá um valor associado a esta. Assim, cada documento poderá ser representado pelo vector de valores de TF-IDF de cada palavra diferente desse documento. Se todos estes vectores que representam documentos forem alinhados, o resultado será uma matriz que tem N linhas, em que N é o número de documentos, e M colunas, em que M é o número de palavras diferentes em todos os documentos.

Seguidamente, transforma-se a questão do utilizador num vector também. Para se calcular este vector terá de ser usada a operação de TF-IDF sobre a questão do utilizador.

Finalmente, como último passo cada vector documento tem de ser comparado com o vector questão do utilizador para que seja encontrado o documento que esteja mais adequado à dada questão do utilizador, como representado na Figura 2.2.

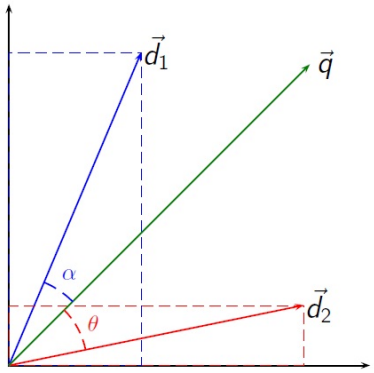

Figura 2.2: Comparação do ângulo entre vectores de documentos e o vector questão.

As técnicas mais comuns para comparar vectores são a distância Euclidiana e a similaridade do cosseno. A distância euclidiana não retorna necessariamente um valor no intervalo $[0, 1]$, enquanto que a função de cosseno retorna um valor normalizado entre $[-1, 1]$. Além disso, o valor da norma da representação do vector em cada um dos eixos é sempre não negativo

visto que é baseada na contagem de palavras. Portanto, o valor retornado pela similaridade do cosseno pertence ao intervalo $[0, 1]$.

$$sim(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}} \tag{2.11}$$

A similaridade do cosseno consiste numa comparação dos dois vectores como descrito pela Fórmula 2.11, em que é representado o produto interno dos vectores por $d_j \cdot q$, e a norma dos mesmos é dada por $\|\mathbf{d_j}\|$ e $\|\mathbf{q}\|$, respetivamente.

2.1.1.4 Métricas Híbridas

Existe ainda um último tipo de métricas de similaridade entre *strings* chamadas métricas híbridas. As métricas híbridas têm por objetivo combinar propriedades dos 3 tipos de métricas enunciados anteriormente.

Por exemplo a similaridade de Monge-Elkan é um algoritmo que segue um esquema de correspondência recursiva (Cohen et al., 2003). A similaridade de Monge-Elkan é baseada no resultado da média de similaridade da divisão das *strings* que se querem comparar.

$$MongeElkan(A, B) = \frac{1}{K} \sum_{i=1}^{K} \max_{j=1}^{L} sim'(A_i, B_j) \tag{2.12}$$

O que faz com que esta medida seja uma métrica híbrida é o facto de esta se basear noutra métrica de similaridade, representada por sim' na Equação 2.12. Uma das medidas mais utilizadas como sim' é a distância de Jaro-Winkler, enunciada anteriormente. Na Equação 2.12, A_i e B_j são sub*strings* das *strings* A e B, respetivamente, L é o número de sub*strings* provenientes de B e K é o número de sub*strings* provenientes de A. Repare-se que a medida de similaridade vai retornar para cada A_i um total de L valores, e que só entra para o somatório aquele que tiver maior valor de similaridade. De notar ainda que qualquer algoritmo de similaridade pode ser usado como sim', desde que esteja normalizado.

Um outro algoritmo que funciona também à base de outro é o algoritmo de *Soft* TF-IDF. Este algoritmo usa os mecanismos básicos do algoritmo TF-IDF, mas enquanto que na versão original eram comparadas somente as palavras que fossem comuns aos documentos, no *Soft* TF-IDF também são consideradas as palavras que sejam similares às que estão a ser consideradas (Cohen et al., 2003), desde que estejam acima de um determinado valor de similaridade. Estes valores de similaridade são baseados em algoritmos anteriormente analisados, como o

Double Metaphone ou a similaridade de Jaro-Winkler. De notar que há normalmente um desconto associado quando é usada uma palavra similar, em vez de uma palavra que seja comum.

2.1.1.5 Comparação entre Diferentes Medidas de Similaridade

Cada um dos quatro tipos principais de medida de similaridade entre *strings* tem a sua importância neste trabalho. No tipo baseado em caracteres, corrigem-se possíveis erros de digitação para que o utilizador não se sinta frustrado quando fizer um pequeno erro de escrita e não haja nenhuma resposta de volta para a sua pergunta. A abordagem baseada em fonemas trata do problema do utilizador não saber como é que uma determinada palavra é escrita. A similaridade baseada em n-gramas ou palavras foca na ordem como uma frase é composta, para que o utilizador possa inserir palavras numa ordem um pouco diferente da ordem usada nas FAQs, e ainda seja encontrada a melhor resposta. Por último, as abordagens híbridas são abordagens que combinam os benefícios das anteriores.

2.1.2 Aprendizagem Automática

A classificação e a regressão são, por ventura, os dois mais importantes problemas no contexto de aprendizagem automática supervisionada. Cada um deles trata da predição de uma resposta b dados os valores do vector de predição a. Tomemos A como o domínio de a e B como o domínio de b. Se b é uma variável que toma valores contínuos mas reais, o problema é chamado regressão. Se B contém dois valores não ordenados o problema é tipicamente referido como classificação binária, e se B envolver mais que duas classes o problema é referido como uma classificação multi-classe.

Em termos matemáticos, estes problemas podem ser reduzidos à tarefa de encontrar a função f(a) que mapeia cada ponto de A a cada ponto de B. A construção de f(a) requer a existência de um conjunto de treino com n observações (i.e. um conjunto de observações que para cada a tem um correspondente b, da forma (a_1,b_1), ... ,(a_n,b_n)).

O objetivo destes sistemas é, portanto, criar uma solução que dando apenas como entrada um vector a, consiga minimizar o número de erros de predição do valor saída b. Esta solução chama-se de classificador.

São agora apresentados alguns dos métodos de aprendizagem automática mais comuns:

2.1.2.1 Classificadores de Vizinhos Mais Próximos:

Classificadores de vizinhos mais próximos (do inglês Nearest Neighbour Classifiers), ou Interpoladores de vizinhos mais próximos (do inglês Nearest Neighbour Interpolators) são duas soluções para problemas de classificação e regressão, respectivamente. Dada uma variável de predição a, estes métodos vão devolver a resposta b, a partir, dos casos de treino que mais se aproximem da variável de predição a. A resposta b é, portanto, decidida por maioria de votos dos vizinhos, ou pela média de valores retornados pelos vizinhos. As contribuições dos vizinhos podem ser ponderadas, por forma a fazer com que a proximidade dos vizinhos seja proporcional ao peso dado.

2.1.2.2 Classificadores Lineares:

Os classificadores lineares, ou modelos de regressão linear, definem a função f(a) em termos da combinação linear de dimensões individuais de uma variável de predição (i.e. características). Os modelos de regressão linear são por muitas vezes usados baseando-se numa abordagem que minimiza o somatório do quadrado dos erros de predição efetuados. Quanto a classificadores lineares, há dois tipos de abordagens, nomeadamente (i) abordagens generativas cujo treino envolve a estimativa de probabilidades dos eventos (e.g., Naive Bayes), e (ii) abordagens discriminativas cujo treino envolve resolver problemas de otimização (e.g. Support Vector Machines).

2.1.2.3 Modelos Baseados em Conjuntos de Árvores:

Os modelos de classificação baseados em árvores de decisão representam a função f(a) como uma árvore de decisão. As árvores de decisão são constituídas por nós que representam decisões e folhas que representam a resposta do classificador. Para se obter a classificação de a começa-se no nó raiz da árvore de decisão. Posteriormente, segue-se o caminho indicado pelas decisões efetuadas em cada nó até chegar a uma folha. As decisões efetuadas em cada nó são baseadas em características retiradas de a. Estas árvores são por sua vez podem ser uniões de outras árvores que encontram a melhor função f(a) para uma determinada porção do domínio de A.

Os algoritmos de que foram usados neste trabalho são todos baseados em árvores, e foram ainda usadas técnicas de *ensemble learning* para combinar várias árvores, tais como *boosting* ou *bagging*.

Boosting consiste numa técnica de aprendizagem automática que iterativamente treina classificadores fracos e junta-os para formar um classificador mais forte (i.e., um classificador que

perante aquele treino responde melhor). *Boosting* iterativamente treina os classificadores com os dados de treino para criar um melhor classificador final. Cada caso de treino tem um peso que lhe está associado, sendo inicialmente um peso igual para todos os casos. A cada iteração o peso dos casos bem classificados diminui e, pelo contrário, o peso dos casos mal classificados aumenta. Assim, *Boosting* foca-se em melhorar os resultados previamente mal classificados. O conjunto final de classificadores fracos, é constituído por um peso associado a cada classificador consoante o seu desempenho nas iterações da fase de treino (i.e., os classificadores fracos que tiveram melhor desempenho contam mais para a classificação do que os que tiveram mais classificações erradas). O algoritmo mais popular de *Boosting* é o algoritmo AdaBoost.

Bagging é uma técnica cujo objectivo é melhorar a exatidão de um modelo de classificação e diminuir a variação do erro. Esta técnica cria um conjunto de dados de treino a partir dos dados de treino originais (i.e., divide os dados de treino em grupos com menos exemplos de treino que repetem muitos exemplos de treino entre cada grupo), treina classificadores a partir de cada um dos elementos do conjunto de dados de treino criado e devolve como resposta o que a maioria dos classificadores devolveu como resposta. Assim, *Bagging* é um algoritmo que cria vários classificadores, a partir de menos dados, e que usa o voto da maioria dos classificadores criados para devolver uma resposta que diminui a variância. Exemplos de *Bagging* são, por exemplo, *Random Forest* e *Rotation Forest*.

2.1.3 Métricas de Avaliação de Sistemas de Classificação

Para se avaliar o desempenho final de sistemas de recuperação de informação e de classificação de documentos, foram no passado propostas várias medidas.

$$\text{Precision} = \frac{|\{\text{documentos relevantes}\} \cap \{\text{documentos devolvidos}\}|}{|\{\text{documentos devolvidos}\}|} \tag{2.13}$$

A Precisão, como descrita pela Equação 2.13, é a relação entre o número de documentos devolvidos pelo sistema que são relevantes, representado por
$|\{\text{documentos relevantes}\} \cap \{\text{documentos devolvidos}\}|$ e o número de documentos devolvidos, representado por $|\{\text{documentos devolvidos}\}|$.

$$\text{Recall} = \frac{|\{\text{documentos relevantes}\} \cap \{\text{documentos devolvidos}\}|}{|\{\text{documentos relevantes}\}|} \tag{2.14}$$

A medida de *Recall*, como descrita pela Equação 2.14 é a relação entre o número de documentos devolvidos pelo sistema que era relevante, representado por

$|\{\text{documentos relevantes}\} \cap \{\text{documentos devolvidos}\}|$ e o número de documentos relevantes, representado por $|\{\text{documentos relevantes}\}|$.

Para relacionar as medidas de Precisão e *Recall* foi proposta a *F-measure*, como mostrado pela Equação 2.15. A relação de 1 para 1 entre Precisão e *Recall* é a F_1 *measure*. Uma relação em que se dá 2 vezes mais enfase à Precisão seria a $F_{0.5}$ *measure*, e uma relação em que se dá 2 vezes mais enfase ao *Recall* seria F_2 *measure*.

$$F_x = \frac{(1 + x^2) \cdot (\text{precision} \cdot \text{recall})}{(x^2 \cdot \text{precision} + \text{recall})} \tag{2.15}$$

A precisão média, dada pela Equação 2.16, é a média de cada uma das precisões nos vários pontos do *Recall*, desde a precisão para 1 documento retornado até à precisão de todos os documentos retornados. Na Fórmula 2.16, a posição na posição k é representada por $P(k)$ e $rel(k)$ é uma função que retorna 1 caso o documento na posição k seja relevante e 0 caso contrário. A *Mean Average Precision* é uma medida que toma em conta não uma só pesquisa mas o conjunto de todos os testes efetuados sobre o sistema, medindo a média da Precisão média.

$$\text{Precisão média} = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{|\{\text{documentos relevantes}\}|} \tag{2.16}$$

2.2 Trabalho Relacionado

A identificação de paráfrases essencialmente corresponde à tarefa de decidir se dois fragmentos de texto, normalmente frases, são paráfrases uma da outra. Esta tarefa não deve ser confundida com extração de paráfrases, que trata de colecionar pares de fragmentos de texto que são paráfrases um do outro, de fontes como a Web (Dolan et al., 2004).

Por ventura, a abordagem mais simples à identificação de paráfrases baseia-se na utilização de recuperação de informação (RI), com uma estratégia que considere o texto como um conjunto de palavras (do inglês *bag-of-words*). Esta estratégia calcula um valor de similaridade do cosseno dado um conjunto de frases, e caso esta similaridade ultrapasse um limite (quer este limite seja empiricamente determinado, ou aprendido a partir dos dados de treino supervisionados), as frases são consideradas paráfrases. Abordagens mais complexas tipicamente envolvem a combinação de várias medidas de similaridade, quer seja usando um método heurístico ou, a partir de aprendizagem automática.

2.2.1 Trabalhos na Área de Identificação de Paráfrases

Por exemplo Kozareva and Montoyo (2006) propuseram uma abordagem de classificação baseada na identificação de paráfrases, combinando características de similaridade a nível lexical e semântico, algumas delas baseadas em conhecidas medidas de sumarização de texto. As experiências com este método revelaram que características simples dependendo apenas em correspondências consecutivas ou em sequência podem identificar um vasto número de paráfrases corretamente, e que combinar múltiplos classificadores também pode ser benéfico. Zia and Wasif (2012) propuseram um melhoramento do conjunto de características, usando heurísticas semânticas (i.e., padrões de negação) para ajudar na deteção de falsas paráfrases.

Zhang and Patrick (2005) propuseram outra abordagem de classificação, em que os pares de frases originais são primeiro convertidos para um texto superficial que aproxima as formas canónicas, a partir de um número limitado de regras de canonização (e.g., mudar as frases da voz passiva para a voz activa, ou mudando diferentes tipos de quantidades numéricas por tokens comuns). O módulo de aprendizagem por árvores de decisão, que usa características de correspondência lexical simples (e.g., a distância de edição entre tokens nas frases em forma canónica), toma como entrada os textos canonizados para o seu processo de aprendizagem supervisionado.

Finch et al. (2005) propuseram um método similar ao que foi usado por Zhang and Patrick (2005) ou por Kozareva and Montoyo (2006), mas usaram métodos de avaliação standard de tradução automática, que são baseados em correspondências de n-gramas ou em distâncias de edição (e.g., BLEU, NIST, WER e PER), como métricas de similaridade entre frases, e usando um classificador do tipo máquina de vectores de suporte (SVM) com kernels de função de base radial para classificar os dados. Os autores usaram a raiz das palavras (Stemming) para juntar palavras relacionadas morfologicamente (i.e., verbos e adjetivos) com a mesma raiz, como um passo de pré processamento para canonizar as frases. Os autores também introduziram o método baseado na medida PER, que aproveita informação de categoria gramatical das palavras contribuindo para as correspondências entre palavras e as não correspondências na frase. Recentemente, Madnani et al. (2012) propuseram ainda uma outra abordagem baseada em métodos de avaliação da área de tradução automática (MT), em função da evolução recente desta área. Estes autores mostraram que um meta-classificador (i.e., um classificador que usa a média não ponderada por três classificadores constituintes para fazer uma decisão final) usando nada mais do que métricas de MT modernas (i.e., 8 diferentes métricas MT, incluindo 6 que não existiam em 2005) supera muitos identificadores de paráfrases recentes que usam mais características de informação linguística.

A abordagem desenvolvida por Qiu et al. (2006) usa um processo com duas fases, também baseado na classificação supervisionada. Ao contrário da maioria dos sistemas de identificação de paráfrases que se foca em similaridade entre frases, Qiu et al. propôs a deteção de diferenças entre frases, fazendo o julgamento de paráfrases baseado na importância destas diferenças. A primeira fase identifica as partes de informação comum ou as unidades chave do conteúdo semântico de cada frase, emparelhando-as entre as frases. Estas partes são tuplos do predicado do argumento (i.e., representações estruturadas do predicado verbal com os seus argumentos, obtidos a partir de um rotulador do papel semântico), que são comparados usando uma técnica de correspondência lexical. Na segunda fase, qualquer parte que não esteja emparelhada é classificada por um modelo SVM como sendo ou não importante. Este classificador SVM usa um conjunto vasto de características representando tuplos não emparelhados, incluindo contagens internas de expressões numéricas, entidades com nome, palavras, papeis semânticos, e se elas são similares aos outros tuplos na mesma frase, assim como características contextuais como o comprimento da frase fonte/alvo e número de tuplos emparelhados. Se as frases não contivessem partes não emparelhadas, ou se todas as partes não emparelhadas fossem classificadas como insignificantes, então as frases eram consideradas paráfrases.

Alguns autores notaram que as paráfrases muitas vezes envolviam o uso de sinónimos ou outras formas das palavras relacionadas e, como tal, os métodos de identificação de paráfrases deviam também ser informados da informação da similaridade ao nível da palavra. Islam and Inkpen (2007) usaram, por exemplo, uma medida baseada na similaridade semântica das palavras (i.e., uma abordagem baseada na segunda ordem de coocorrência de informação mútua pontual, anteriormente descrita por Islam and Inkpen (2006)), juntos com normalização e versões modificadas do algoritmo de correspondência de strings baseados na subsequência comum mais comprida (LCS, do inglês Longest Common Subsequence) (i.e., três versões diferentes do LCS, tendo uma soma ponderada dos seus valores). Um procedimento de aprendizagem supervisionada foi usado para definir o limite sobre o qual as frases eram consideradas paráfrases.

Mihalcea et al. (2006) e Fernando and Stevenson (2008) apresentaram ambos algoritmos para identificação de paráfrases que fazem uso da informação de similaridade entre palavras, derivada do WordNet, nas métricas de similaridade de frases consideradas. Mihalcea et al. (2006) usou medidas de similaridade palavra a palavra (e.g., métricas baseadas em conhecimento que usam o WordNet, ou outras medidas baseadas na análise de um corpus, nomeadamente informação mútua pontual e análise de semântica latente) em conjunto com uma medida de especificidade da palavra para estimar a similaridade semântica entre os pares de frases. Uma das funções heurísticas usadas em combinação com as métricas era alinhar pares de palavras de acordo com a sua similaridade máxima, e pesando as palavras alinhadas de acordo com a

sua especificidade. Um limite de 0.5 foi usado na classificação (i.e., pares com uma pontuação acima do limite eram classificados como paráfrases). Fernando and Stevenson (2008) propuseram uma abordagem de matriz de similaridade em que todas as similaridades palavra a palavra eram tomadas em conta, e não só as que têm máxima similaridade entre as frases, como no método proposto por Mihalcea et al. (2006). Eles experimentaram com 6 métricas de similaridade do WordNet diferentes para popular a matriz de similaridade, em que esta é enão usada para processar a similaridade entre as representações vectoriais das frases.

Rus et al. (2008) propuseram um método de identificação de paráfrases baseado num grafo de subsunção léxico-sintática, contando na informação lexical, sintática, sinonímia e antonímia. A informação sinonímia e antonímia é extraída do WordNet. Nesta abordagem, grafos são usados para modelar a informação linguística embebida em ambas as frases, com vértices a representar conceitos e arestas a representar relações sintáticas entre os conceitos. As duas frases são paráfrases se cada um dos grafos de hipóteses agrupar o outro. O algoritmo de agrupamento começa por encontrar um isomorfismo entre os conjuntos de vértices associados a ambas as frases, usando WordNet para encontrar sinónimos. Então, o algoritmo verifica se as arestas classificadas também têm correspondências, considerando a relação equivalente entre fenómenos linguísticos como os possessivos ou negações (e.g., usando antónimos do WordNet para lidar com as negações). A pontuação final do agrupamento é a soma ponderada de cada pontuação dos correspondentes vértices e arestas, com pesos descobertos a partir de regressão linear.

Socher et al. (2011) propôs uma abordagem que incorpora as similaridades entre as características das palavras e as características de múltiplas palavras extraídas dos nós das árvores sintáticas. Esta abordagem envolve dois componentes principais, nomeadamente (i) um desenrolar recursivo do auto-encoder para aprendizagem das características de forma não supervisionada a partir das árvores de análise não classificadas, e (ii) uma camada de agrupamento dinâmico que devolve uma representação de tamanho fixo. Este auto-encoder recursivo é uma Rede Neuronal Artificial recursiva que aprende as representações das características para cada nó da árvore, de modo que o vector palavra filho de cada nó possa ser recursivamente reconstruído. Estas representações de características são usadas para processar a matriz de similaridade que compara tanto as palavras uma a uma como todos os nós de características não terminais de ambas as frases. A camada de agrupamento dinâmico é usada para manter tanta informação global resultante desta comparação quanto possível, e para colmatar com o problema da diferença de comprimento entre as duas frases. Um classificador Softmax, contando com as características baseadas na matriz de similaridade em conjunto com características simples como, por exemplo, o comprimento da frase, ou a percentagem de palavras numa frase que estão na outra frase e vice-versa, é finalmente usado para classificar se duas frases são

paráfrases ou não.

Blacoe and Lapata (2012) também abordaram o problema da identificação de paráfrases através de uma modelação do significado composicional das frases, usando métodos de word clustering para captura da similaridade das palavras. Os autores experimentaram várias combinações possíveis de representações de palavras (e.g., um simples espaço semântico onde o vector palavra representa a sua coocorrência entre palavras vizinhas, um espaço sintático baseado na distribuição ponderada de tuplos que codificava as relações as relações de coocorrência entre palavras, e incorporações de palavras calculadas através de um modelo de linguagem neural) e métodos de composição para representar frases usando os vectores das suas palavras constituintes (e.g., baseados em adição de vectores, multiplicação, e redes neuronais artificial recursivas). Para cada um dos três vectores de origem e para cada um dos três métodos de composição, os autores criaram as seguintes características: (a) um vector representando um par de frases originais quer seja por via da concatenação ou da subtração; (b) um vector codificando que palavras aparecem nas frases; e (c) um vector criado a partir das seguintes informações: a similaridade do cosseno entre os vectores frase, o comprimento de S_1, o comprimento de S_2, e a sobreposição de unigramas entre as duas frases. Então, eles usaram um classificador liblinear introduzido por Fan et al. (2008) para classificar pares de frases como sendo paráfrases ou não. Os resultados experimentais mostram que abordagens superficiais (i.e., simples representações do espaço semântico) são tão boas como as alternativas que são computacionalmente intensivas.

A introdução de métricas de similaridade baseadas em características derivadas de dependências entre árvores sintáticas também foi mostrada como sendo um melhoramento na performance, sob a assumpção que duas frases que são paráfrases, devem ter árvores de dependência que são próximas uma da outra. Por exemplo, Wan et al. (2006) usou características baseadas em métricas de tradução automática (i.e., características baseadas no BLEU) em combinação com várias outras características baseadas nas relações de dependência e árvores de distância de edição, num classificador SVM. Recentemente, Das and Smith (2009) usaram gramáticas de dependência quasi-síncronas para modelar a estrutura das frases envolvidas na comparação e nas correspondências. Em suma, estes autores usaram um modelo gerador que gera a paráfrase para a frase dada, e mais tarde usa inferência probabilística para decidir se as duas frases partilham a relação com a paráfrase. O modelo incorpora semântica lexical e sintática a partir do formalismo das gramáticas de dependência quasi-síncronas, que estabelecem ligações fracas entre as estruturas sintáticas das duas frases, assim, dando algum espaço a algumas divergências. Usando um produto de peritos, os autores também combinaram o modelo proposto com um modelo de regressão logística, usando algumas características de sobreposição lexical

Referência	Breve Descrição em Inglês	Algoritmo	Acc.	F_1
Mihalcea et al. (2006)	Baseline with cos similarity and TF-IDF weights	unsupervised	0.654	0.753
Zhang and Patrick (2005)	Lexical features after text canonicalization	supervised	0.703	0.795
Mihalcea et al. (2006)	Combination of word-to-word similarities	unsupervised	0.703	0.813
Rus et al. (2008)	Graph subsumption	unsupervised	0.706	0.805
Qiu et al. (2006)	Sentence dissimilarity classification	supervised	0.720	0.816
Islam and Inkpen (2007)	Combination of semantic and string similarity	unsupervised	0.726	0.813
Blacoe and Lapata (2012)	Semantic spaces from word clustering	supervised	0.730	0.823
Fernando et al. (2008)	Wordnet metric and vector similarity	unsupervised	0.741	0.824
Zia and Wasif (2012)	Semantic heuristic features	supervised	0.747	0.818
Finch et al. (2005)	Combination of MT evaluation measures	supervised	0.750	0.827
Wan et al. (2006)	Dependency-based features	supervised	0.756	0.830
Das and Smith (2009)	Product of experts, using dependency parsing	supervised	0.761	0.827
Kozareva and Montoyo (2006)	Combination of lexical and semantic features	supervised	0.766	0.796
Socher et al. (2011)	Recursive autoencoder with dynamic pooling	supervised	0.768	0.836
Madnani et al. (2012)	Modern machine translation metrics	supervised	0.774	0.841

Tabela 2.1: Comparação de diferentes métodos que foram propostos anteriormente.

que são populares nestes trabalhos.

2.2.2 Trabalhos na Área de Pesquisa em FAQs

A Pesquisa em FAQs tem por objectivo encontrar uma pergunta específica numa base de dados de FAQs. Estas pesquisas têm de identificar as perguntas que têm o mesmo significado que a frase de pesquisa.

SMS based FAQ Retrieval, cujo objetivo é avaliar sistemas que recebam uma SMS com uma questão, encontrem uma questão semelhante num corpora de FAQs, e devolvam a melhor resposta à SMS. Este concurso estava projetado para ser resolvido nas línguas Inglesa, Hindi e Malaia.

Este concurso tinha um fator que complicava o problema nomeadamente o facto de poder ser usada ”linguagem de SMS”, ou seja, abreviaturas, frases que não estavam bem estruturadas, omissão de palavras, palavras mal escritas, entre outros problemas.

Os vencedores foram um grupo da Universidade de Iowa com um sistema que, caso a questão estivesse em Inglês usava o *Google Spelling Suggestions* para completar palavras. Seguidamente, faziam as SMSs passar por um tradutor que tinha uma tabela de abreviações e ”traduzia”a mensagem para Inglês mais formal. Esta tabela foi criada manualmente. Num passo seguinte, usava-se um corretor automático na SMS traduzida para eliminar algumas palavras que eram pequenas abreviações, ou acrónimos, ou mesmo palavras mal escritas. Por último, usaram também uma noção de *Term Frequency* para identificar o que era mais importante na pergunta escrita em linguagem SMS.

2.3 Sumário e Discussão Crítica

Neste capítulo, foram visitados os tipos mais comuns de abordagens de métricas de similaridade. Cada um dos tipos de métricas apresentados representa diferentes aspectos das frases que são tomados em conta na nossa abordagem para resolver o problema. De seguida, foi apresentado o conceito de aprendizagem automática. Por último, os conceitos de algumas métricas de Avaliação usadas pelos Sistemas de Classificação foram apresentados para uma avaliação mais completa da abordagem tomada.

A Tabela 2.1 apresenta uma visão global das diferentes abordagens que foram usadas, em conjunto com os resultados correspondentes, sobre o Microsoft Research Paraphrase Corpus, um conhecido conjunto de dados usados como referência na área de deteção de paráfrases, introduzido por Dolan et al. (2004). É mostrado que no actual estado-da-arte a abordagem que funcionou melhor usava métricas de tradução automática modernas.

Capítulo 3

Método Proposto

No capítulo 3 é apresentado o sistema desenvolvido para a deteção de paráfrases, o qual foi posteriormente usado para a realização de experiências. Este capítulo está, assim, dividido em duas secções. Na primeira secção é apresentada a arquitetura geral do sistema. Na segunda secção são apresentadas as características usadas nos modelos de classificação.

3.1 Arquitetura Geral

O nosso trabalho toma a abordagem padrão de modelar a identificação de paráfrases como um problema de classificação supervisionado, como é mostrado na Figura 3.3. A classificação entre um par de frases devolve como resposta uma de duas classes, paráfrase ou não paráfrase.

Para extrair os vectores de características usámos tanto bibliotecas de algoritmos de medidas de similaridade entre *strings* já existentes, como bibliotecas para extração de diferentes representações das frases.

A conhecida biblioteca de processamento de língua natural MorphAdorner[1] foi usada como forma de efetuar pré-processamento das frases, devolvendo a *tokenização*, *stemming*, e *lematização* (i.e., um processo de redução das palavras para a sua raiz lexical que é mais avançada do que *stemming*, que reduz os verbos para o infinitivo, e que também faz alguma uniformização ortográfica).

Usamos a biblioteca Apache Commons[2] para o pré-processamento das frases, devolvendo os valores das chaves fonéticas de Double Metaphone.

[1] `http://morphadorner.northwestern.edu/`
[2] `http://commons.apache.org/proper/commons-codec/`

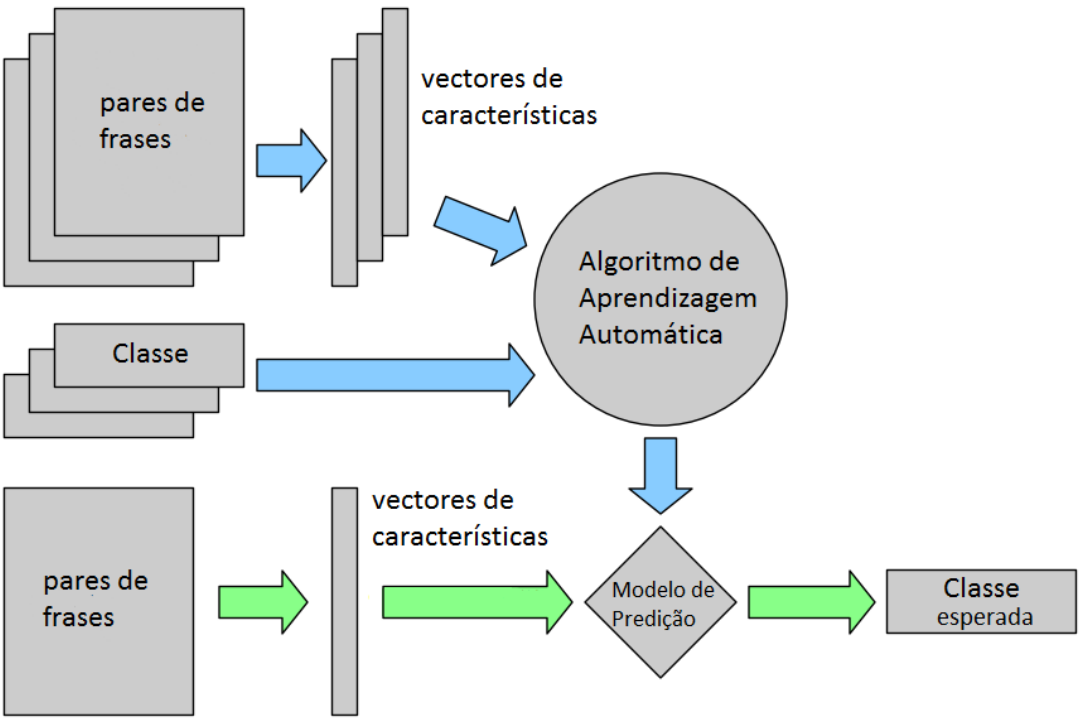

Figura 3.3: Método de aprendizagem automática usado na identificação de paráfrases.

É ainda usada a biblioteca Second String[1] para criar os trigramas de caracteres. Os trigramas de caracteres são conjuntos de três caracteres extraídos da frase original.

Uma das características usadas nos vectores de características são códigos representativos de cada aglomerado de palavras (i.e., palavras com significado semelhante). Estes códigos representativos são chamados de *word clusters*. Retiramos o código dos *word clusters* segundo o algoritmo explicado em 5.2. Foram usados como *word clusters*, os *Brown Clusters*[2]. Os *Brown Clusters* são criados a partir de um modelo de linguagem baseado em classes hierárquicas (Turian et al., 2010). Na Figura 3.4 temos um exemplo de uma árvore hierárquica construída com base nas palavras *CEO*, *chairman*, *president*, *November*, *October*, *run*, *sprint* e *walk*. Os código das palavras são retirados segundo o caminho efetuado desde a raiz da árvore até à folha que tem a palavra que se quer (e.g., para a palavra *president*, o código é 0011). A árvore é hierárquica pois todas as palavras com um determinado prefixo pertencem à mesma classe mais genérica (e.g., se tomarmos, por exemplo, as palavra *run* com código 100, *sprint* com

[1] http://secondstring.sourceforge.net/
[2] http://metaoptimize.com/projects/wordreprs/

código 101, e *walk* com código 11, notamos que todas as palavras deste conjunto com código que comece por 10 são verbos que implicam corrida e que todas as palavras com código que comece por 1 são verbos que exprimem diferentes formas de movimentação).

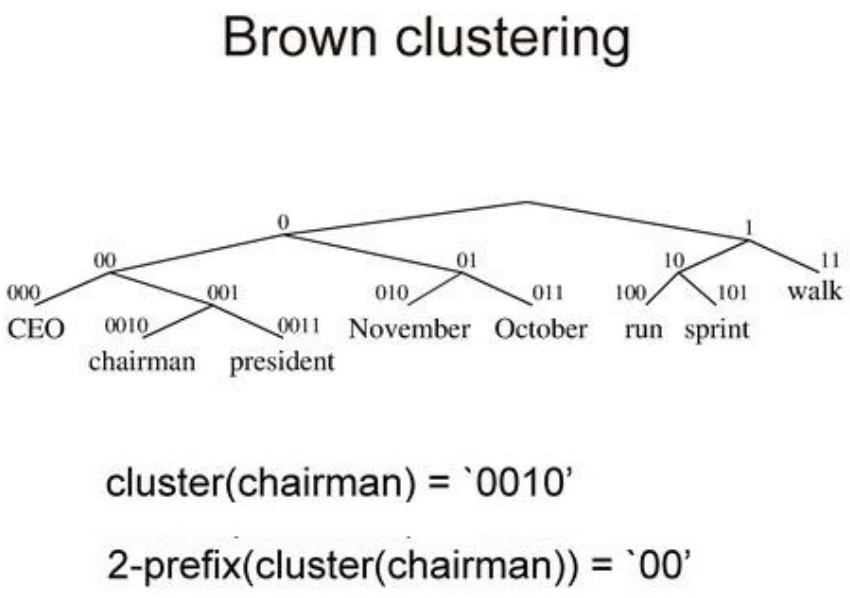

Figura 3.4: Exemplo de *Brown Clusters*.

Experimentámos diferentes algoritmos de aprendizagem automática baseados em conjuntos de árvores (i.e., AdaBoost, Random Forests e Rotation Forests), para combinar um vasto conjunto de características baseadas em informação lexical. Em termos dos métodos de aprendizagem considerados, escolhemos experimentar modelos baseados em conjuntos de árvores devido à superior performance mostrada por estes métodos em vários problemas de aprendizagem automática (Rokach, 2010). Usamos as implementações de aprendizagem automática através da ferramenta Weka para os modelos de classificação[1] que considerámos, e mantivemos os parâmetros predefinidos pela ferramenta para todos os métodos (i.e., AdaBoost, Random Forests e Rotation Forests).

O algoritmo de AdaBoost cria um conjunto de árvores ao gerar e chamar classificadores fracos (i.e., *decision stumps*) numa série de rondas (Freund and Schapire, 1997). Para cada chamada, a distribuição dos pesos pelos exemplos de treino é actualizada, indicando a importância

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

dos diferentes exemplos para a classificação. A cada ronda, os pesos dos exemplos que foram incorretamente classificados são aumentados, e os pesos de cada exemplo corretamente classificado diminuem, assim o novo classificador foca-se nos exemplos que até então não foram corretamente classificados. O modelo final é uma combinação de múltiplos classificadores fracos, e a classificação é dada com base na maioria dos votos ponderados.

Random Forests funcionam construindo um conjunto de árvores de decisão enquanto se está na fase de treino, devolvendo a classe que for a moda das classes retornadas pelas árvores individualmente (Breiman, 2001). O método combina as ideias de *bagging* (i.e., escolher um conjunto de treino diferente para cada árvore, tomando n exemplos do conjunto de N casos de treino disponíveis) e seleção aleatória das características (i.e., para cada nó de cada árvore, nós aleatoriamente escolhemos m variáveis, leia-se características, para decidir nesse nó).

Rotation Forests usam o mesmo princípio por trás de Random Forests, mas cada árvore é treinada sobre todos os dados num espaço de características rotativo (Rodriguez et al., 2006). Amostras de inicialização são retiradas do conjunto de treino para classificadores individuais, como num comum uso de *bagging*. A heurística principal é aplicar a extração de características e reconstruir, subsequentemente, um conjunto completo de características para cada classificador num conjunto de árvores. Para tal, o conjunto de características é dividido aleatoriamente em K subconjuntos, o componente principal de análise (PCA) é então corrido separadamente para cada subconjunto. Um novo conjunto de n características extraídas linearmente é construído por agrupamento de todos os PCA, e cada classificador é treinado com um destes conjuntos de dados.

3.2 Características Usadas no Classificador

Quanto às características consideradas, foram usados diferentes tipos de métricas de similaridade envolvendo diferentes representações do conteúdo lexical das frases. Foram especificamente consideradas representações das frases baseadas em:

1. *Tokens* extraídos de versões em letras minúsculas das frases originais;
2. *Lemmas* extraídos de versões em letras minúsculas das frases originais;
3. *Stems* extraídos de versões em letras minúsculas das frases originais;
4. *Word clusters* associados aos *tokens* extraídos das frases, usando um total de 100 *clusters* diferentes;

5. *Word clusters* associados aos *tokens* extraídos das frases, usando um total de 320 *clusters* diferentes;
6. *Word clusters* associados aos *tokens* extraídos das frases, usando um total de 3200 *clusters* diferentes;
7. Chaves fonéticas de Double Metaphone (Philips, 2000) associadas aos *tokens* extraídos de versões em letras minúsculas das frases originais;
8. Trigramas de caracteres extraídos das frases originais;

Nos Items 4, 5 e 6 da enumeração anterior, considerámos os *word clusters* obtidos através do procedimento descrito por Turian et al. (2010), com base num longo corpus de textos jornalísticos. Quando se gerou as representações de frases baseadas em *word clusters*, inicialmente tentámos usar *clusters* associados aos termos exactos que se encontravam presentes nas frases e, no caso dos *tokens* não estarem presentes no corpus de textos jornalísticos, nós também tentámos obter o *cluster* correspondente ao *lemma* ou ao *stem* associado ao *token* correspondente.

As oito diferentes representações que foram descritas anteriormente foram usadas para gerar características com base (i) na similaridade do cosseno entre vectores de características, considerando os pesos de TF-IDF para cada característica, e (ii) no coeficiente de similaridade de Jaccard entre conjuntos de características. Para lidar com pequenas variações em termos de ortografia das palavras, para além do uso do algoritmo de Double Metaphone, processamos uma característica baseada na métrica de similaridade de Soft TF-IDF, que mede a similaridade entre representações vectoriais das frases, como o caso da métrica de similaridade do cosseno, mas considerando uma métrica interna de similaridade para encontrar palavras equivalentes. As características baseadas em Soft TF-IDF foram processadas com base nas representações 1,2,3 da enumeração anterior, usando a métrica de similaridade de Jaro-Winkler entre palavras com um limite de 0.9, como medida interna de similaridade.

Também são processadas características baseadas no comprimento da frase original (i.e., duas características, cada uma codificando o comprimento de cada frase, e uma terceira característica contendo a diferença absoluta entre os comprimentos das frases), e o comprimento da subsequência comum mais longa (do inglês *Longest Common Subsequence*) entre as frases. No processamento do comprimento da subsequência comum mais longa, consideramos frases correspondentes a todas as representações dadas excepto pelo último item na enumeração das representações anterior (i.e. as novas frases obtidas a partir das originais, depois de substituir os seus tokens (i) pelas suas versões em letras minúsculas, (ii) pelos correspondentes *lemmas*, (iii) *stems*, (iv, v and vi) *word clusters*, e (vi) chaves de Double Metaphone).

Foi processada a distância normalizada de compressão como uma métrica de similaridade entre frases, usando todas as representações excepto a última. A distância de compressão normalizada é baseada na pesquisa da complexidade de Kolmogorov, e a ideia básica é se temos duas strings x e y, com alguma sobreposição entre elas, então a concatenação e compressão das strings $x + y$ deveria ser menor do que a concatenação strings comprimidas separadamente x e y (Li et al., 2004).

Por último, também foram calculadas características baseadas em métricas comuns na área de tradução automática, nomeadamente (i) BLEU, (ii) METEOR, (iii) WER e (iv) TER. As características correspondentes a estas quatro métricas são obtidas a partir da representação de frases correspondendo a todos os items excepto o último item da enumeração anterior.

Em suma, o BLEU é a métrica mais usada nas avaliações MT (Papineni et al., 2002). É calculada como a quantidade de sobreposições de n-gramas, para diferentes valores de n (i.e., no nosso caso usámos BLEU-3, BLEU-6 e BLEU-9), entre as duas frases, normalizadas pelo facto de haver uma penalidade associada ao comprimento da frase. METEOR é uma variação do BLEU que usa uma combinação de *precision* e de *recall*, ao contrário de BLEU que se foca na *precision* (Lavie and Banerjee, 2005). Quanto ao *Word Error Rate* (WER), é uma métrica simples baseada em programação dinâmica que é definida como o número de edições necessárias para transformar uma string noutra. A métrica conhecida como *Translation Error Rate* (TER) difere do WER na medida em que inclui um algoritmo de heurística para tratar com trocas em adição às inserções, eliminações e substituições (Snover et al., 2006). Embora algumas destas métricas de avaliação de MT possam diretamente considerar o uso de recursos lexicais exteriores (e.g. sinónimos do WordNet), nós usamos versões mais simples das métricas, e calculamos sobre diferentes representações das strings originais (e.g., usar *tokens* do texto em letras minúsculas, *lemmas*, *stems*, *word clusters*, e chaves de Double Metaphone).

Os algoritmos para devolver medidas de similaridade a partir das diferentes representações das frases para criar os vectores de características foram escolhidos para formarem um conjunto completo de avaliação dos conjuntos de frases recebidos. Assim, usamos (i) implicitamente uma métrica baseada em caracteres na medida de Soft TF-IDF, (ii) implicitamente métricas baseadas em fonemas visto que uma das representações das frases é o conjunto de chaves fonéticas de Double Metaphone, (iii) implicitamente métricas baseadas em n-gramas com o uso de representações das frases em trigramas de caracteres, (iv) explicitamente métricas baseadas em palavras com o uso de medidas como o TF-IDF, (v) explicitamente métricas híbridas usando Soft TF-IDF e (vi) explicitamente medidas de similaridade extraídas directamente de algoritmos de tradução automática. Usamos a biblioteca SecondString para criar os valores das medidas de similaridade TF-IDF, Jaccard e Soft TF-IDF.

No total, o modelo considerou 79 características diferentes representando cada par de frases.

3.3 Sumário

Neste capítulo foi descrita a abordagem que foi tomada neste trabalho. Também foi feita uma descrição de como os conceitos fundamentais se ligaram com a abordagem feita.

Capítulo 4

Validação Experimental

Este capítulo descreve a metodologia usada na avaliação do método proposto, assim como os resultados obtidos.

4.1 Metodologia de Avaliação

Nas nossas experiências, usámos o Microsoft Research Paraphrase Corpus (MSRPC) introduzido por Dolan et al. (2004). O corpus consiste num vasto conjunto de pares de frases $\{S_1, S_2\}$, previamente rotulados indicando se as frases são paráfrases ou não. Os dados do MSRPC contêm 5,801 pares de frases, e foi feita sobre este uma divisão de 4,076 pares de treino (dos quais 67.5% são paráfrases) e 1,725 pares de teste (dos quais 66.5% são paráfrases).

Brockett and Dolan (2005) mencionam que este corpus for criado de uma forma semi-automática primeiro treinando um classificador SVM num conjunto de pares de frases com 10,000 frases anotadas, e que, posteriormente, aplicando o SVM em 49,375 pares de frases não anotadas, devolvendo os pares de frases com tendências a uma identificação excessiva, i.e., gerando algumas paráfrases que são falsas. Um total de 5,801 destes 49,375 pares foram aleatoriamente escolhidos e apresentados a um júri humano para afinação entre verdadeiras e falsas paráfrases.

Uma estrita definição de paráfrase forçaria os dois textos candidatos a terem um significado idêntico. No entanto, os criadores do MSRPC acharam que esta definição rígida iria limitar os pares de paráfrases a serem, virtualmente, cópias uma da outra. Tais pares devolvem dados interessantes para analisar alterações menores quer a nível sintático, quer a nível lexical. No entanto, os autores de MSRPC queriam capturar diferenças mais interessantes e complexas.

Isto levou a uma definição menos rígida de paráfrase sendo antes *implicação bidirecional completa*, e um total de 3,900 pares que foram marcados pelo júri humano como tendo *a maioria das implicações bidirecionais*. Ainda assim, a mensagem principal das diretrizes para os anotadores do corpus poderem considerar duas frases paráfrases era que pares de frases deviam descrever o mesmo evento e conter a mesma informação relevante para esse mesmo evento. Cada frase era classificada primeiro por dois membros do júri, que tiveram uma média de concordância na classificação de 83%, e um terceiro membro do júri resolveria conflitos. Esta média de concordância na classificação pode ser considerada como um limite superior da exatidão que pode ser obtida com métodos automáticos.

Nas nossas experiências também usámos os dados dos concursos FireFAQ 2011 e FireFAQ 2012. Ambos os concursos têm a mesma estrutura, ou seja, ambos têm uma base de dados de FAQs, um conjunto de perguntas de treino e um conjunto de perguntas de teste. Tanto no FireFAQ 2011 como no FireFAQ 2012, é usada a mesma base de dados de FAQs que contém 7,251 perguntas com as respectivas respostas associadas.

No caso do FireFAQ 2011, o corpus de treino tinha 701 perguntas com o mesmo significado na base de dados de FAQs e 370 perguntas que não tinham uma pergunta com o mesmo significado na base de dados de FAQs. O corpus de teste tem 728 perguntas que tinham uma pergunta com o mesmo significado na base de dados de FAQs e 2677 perguntas que não tinham uma pergunta com o mesmo significado na base de dados de FAQs.

No caso do FireFAQ 2012, o corpus de treino tinha 1429 perguntas com o mesmo significado na base de dados de FAQs e 3047 perguntas que não tinham uma pergunta com o mesmo significado na base de dados de FAQs. O corpus de teste tem 726 perguntas que tinham uma pergunta com o mesmo significado na base de dados de FAQs e 1007 perguntas que não tinham uma pergunta com o mesmo significado na base de dados de FAQs.

4.2 Resultados

No primeiro conjunto de testes, analisámos a performance obtida com diferentes algoritmos de aprendizagem e um conjunto de características completo como descrito na Capítulo 3. A Tabela 4.2 apresenta os resultados obtidos, mostrando que o classificador Rotation Forest obteve melhores resultados.

Num segundo conjunto de testes, tentámos analisar os méritos relativos de características diferentes, a partir de estratégias de seleção de características *greedy-forward* e *greedy-backward*

Algoritmo	Accuracy	F_1
AdaBoost	0.704	0.767
Random Forest	0.748	0.825
Rotation Forest	0.773	0.837

Tabela 4.2: Resultados com diferentes métodos de aprendizagem automática.

Estratégia	Accuracy	F_1	Num. de Características
Forward	0.752	0.826	8
Backward	0.773	0.837	79

Tabela 4.3: Resultados da seleção de características.

em conjunto com um modelo de classificação baseado em Rotation Forests. A seleção de características *greedy-forward* adiciona uma característica de cada vez ao conjunto de características já selecionadas, e verifica quão eficaz é uma característica ao treinar & testar o classificador. A melhor característica é então adicionada ao conjunto de características já selecionados e este processo termina quando adicionar outra característica não cria um melhor classificador. A eliminação de características *greedy-backward* é um procedimento similar, em que se começa com um conjunto de características completo e, uma a uma, se retira as características que menos contribuem para a exatidão da classificação, até que a qualidade dos resultados baixe abaixo de um limite de aceitação. A Tabela 4.3 mostra os resultados obtidos, apresentando o número de características que foi guardado por cada estratégia, juntos com os resultados de exatidão e F_1.

As 8 características selecionadas pela estratégia *Forward* incluem os valores de TF/IDF e Soft TF/IDF calculados com os *tokens* em letra minúscula, o *longest common subsequence* calculado com base em *word clusters*, e métricas de tradução automática calculadas com base em *lemmas*, *stems* e *word clusters*. A característica mais discriminativa e importante foi o TF/IDF.

Das características selecionadas podemos concluir que (i) as frases tinham poucas variações de escrita e que o valor de usar chaves de Double Metaphone foi reduzido, (ii) cinco das oito características selecionadas pelo procedimento *Greedy-Forward* eram métricas de tradução automática, (iii) sete das oito características são valores que têm origem em diferentes algoritmos, e (iv) TF-IDF (ou Soft TF-IDF) são os melhores algoritmos individualmente para analisar se duas frases são ou não paráfrases uma da outra.

Analisámos manualmente algumas fontes de erro na identificação de paráfrases. A Tabela 4.4 apresenta alguns dos pares de frases que foram incorretamente classificados. Os primeiros dois pares mostram que há frases no corpus que têm palavras muito similares e que aparentemente se referem à mesma entidade e eventos mas, ainda assim, são consideradas como MSRPC como não sendo paráfrases. Os últimos dois exemplos mostram que há problemas na

Frase1	Frase2	é Paráfrase	Avaliação
Crews worked to install a new culvert and prepare the highway so motorists could use the eastbound lanes for travel as storm clouds threatened to dump more rain.	Crews worked to install a new culvert and repave the highway so motorists could use the eastbound lanes for travel.	falso	verdadeiro
Bethany Hamilton remained in stable condition Saturday after the attack Friday morning.	Bethany, who remained in stable condition after the attack Friday morning, talked of the attack Saturday.	falso	verdadeiro
Remaining shares will be held by QVC's management.	Members of the QVC management team hold the remaining shares.	verdadeiro	falso
Mr. Malik assured him that he would be considered a martyr if he did not return, the witness testified.	Mr. Malik assured him that he would be considered a martyr if anything happened to him as a result of his trip, the witness said.	verdadeiro	falso

Tabela 4.4: Exemplos de frases avaliadas incorretamente.

identificação de paráfrases quando estas estão em vozes gramaticais diferentes ou quando há palavras diferentes.

Para além de termos feito testes usando o Microsoft Research Paraphrase Corpus, nós também fizemos alguns testes com dados de uma aplicação de domínio prático, onde a identificação de paráfrases é um fator chave, nomeadamente com os dados de mono-lingual English da tarefa do FireFAQ em relação a FAQ retrieval (Contractor et al., 2013). O objectivo desta tarefa é encontrar a questão Q de um corpus de FAQs com a melhor correspondência à questão com ruído S (i.e., a tarefa considera consultas às bases de dados com perguntas FAQ que estão escritas em linguagem *SMS*, onde os utilizadores podem comprimir texto quer seja por omissão de letras, uso de gíria, etc.). Desta forma, modelámos a tarefa da mesma forma que a identificação de paráfrases, usando um classificador que combina múltiplas métricas de similaridade entre questões SMS S e questões Q do corpus de FAQs. Retornamos até cinco questões que foram consideradas paráfrases, ordenadas de acordo com um valor de confiança retornada pelo classificador.

Para tratar de gíria específica de SMS, nós usamos um passo de pré-processamento de normalização de linguagem SMS, em que uma lista de 2561 abreviações SMS, retiradas da Web, foram usadas para substituir palavras da gíria SMS por equivalentes palavras em inglês regular (e.g., ”btw”é substituído por ”by the way”). Mensagens *SMS* podem também ter erros tipográficos ou de digitação (e.g., devido ao pequeno tamanho das teclas em telefones móveis), e nós tentamos endereçar este fator através do uso de métricas de similaridade que usam o algoritmo de Double Metaphone, que já foi considerado nos testes com o MSRPC. Foi usado o conjunto de características da Secção 3 tal como as representações descritas nessa secção, depois de um pré-processamento inicial em que toda a linguagem *SMS* foi substituída.

A Tabela 4.5 apresenta os resultados obtidos com os dados das competições FireFAQ 2011 e

	P@1	P@5	MRR
FireFAQ 2011	0.879	0.899	0.981
FireFAQ 2012	0.725	0.818	0.943

Tabela 4.5: Resultados sobre os dados do concurso FireFAQ.

2012, quando se considera um modelo baseado num classificador Rotation Forest, com todas as características introduzidas na Secção 3 mais as características que têm em conta a linguagem SMS. A avaliação de resultados é apresentada em termos de três métricas, nomeadamente a média de precisão nas posições 1(P@1) e 5(P@5) (i.e., ver se a questão correta das FAQ é devolvida como resultado de topo, ou se este é devolvido num dos 5 resultados de topo), e a Média de Rank Recíproco (MRR) em que é encontrada a resposta certa, de entre as 5 perguntas que são retornadas.

Os resultados na Tabela 4.5 mostram que a nossa abordagem baseada em deteção de paráfrases consegue bons resultados.

Na Figure 4.5 mostramos os diferentes resultados dos participantes no concurso FireFAQ 2011 na tarefa de pesquisa de perguntas na língua Inglesa com pesquisas escritas na língua Inglesa. Tendo no eixo das abcissas os 13 participantes do concurso e no eixo das ordenadas os resultados de MRR.

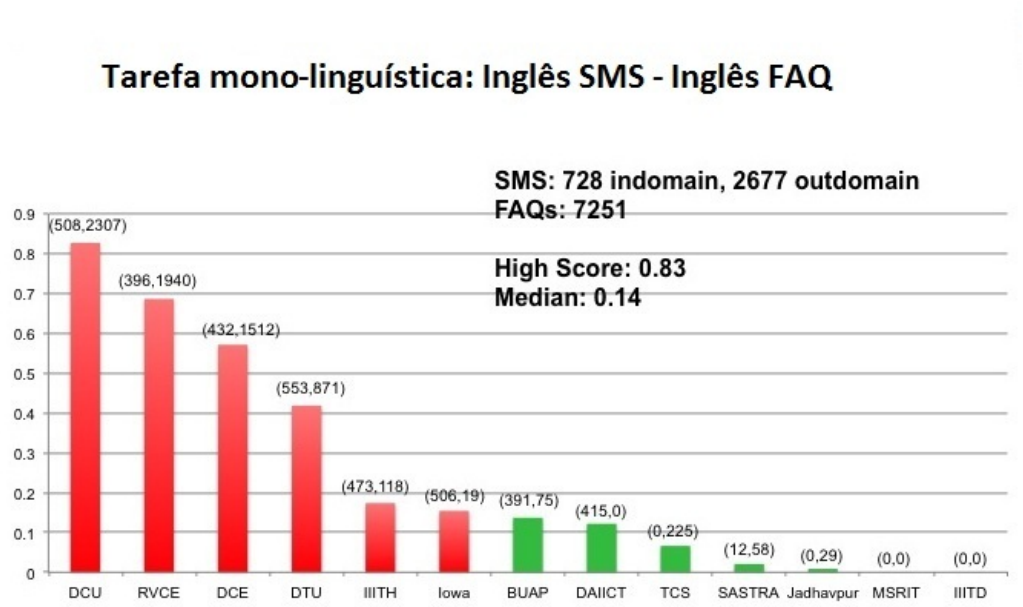

Figura 4.5: Resultados de todos os participantes no concurso FireFAQ 2011, na tarefa mono-línguistica de pesquisa em FAQs de Inglês para Inglês.

Na concurso FireFAQ 2011, o sistema participante que conseguiu a melhor pontuação de MRR obteve 0.896, um resultado ligeiramente inferior em comparação com o nosso. A mediana da pontuação de MRR foi de 0.14. O sistema participante que conseguiu melhores valores de MRR identificaram 494 perguntas correspondentes às pesquisas de teste na base de dados FAQ, e corretamente identificaram 2311 pesquisas como não tendo uma pergunta correspondente, enquanto que o nosso método obteve 509 pesquisas corretas com correspondências à base de dados e corretamente identificou 2562 pesquisas como não tendo uma pergunta correspondente na base de dados de FAQs.

4.3 Sumário e Discussão

Neste capítulo foram descritos os dados que foram usados nas nossas experiências. Foram, também, apresentados todos os resultados obtidos e uma comparação com os resultados obtidos por outros trabalhos sobre os mesmos corpus.

Capítulo 5

Conclusões e Trabalho Futuro

Propusemos um método de identificação de paráfrases baseado em aprendizagem automática supervisionada, que envolve treinar um classificador que usa características correspondentes a métricas de similaridade entre strings que dependem da informação lexical. Esta dissertação reportou sobre um conjunto de experiências que usaram o conhecido Microsoft Research Paraphrase Corpus, em que conseguimos uma classificação com 0.77 de exatidão e uma medida F1 de 0.84. Desta forma, é demonstrado que algoritmos de aprendizagem automática usando características relativamente simples podem obter resultados semelhantes aos resultados do actual estado-da-arte nesta tarefa. Usando os dados do concurso internacional FireFAQ, mostramos que uma abordagem de deteção de paráfrases pode ser uma boa abordagem para encontrar uma pergunta com o mesmo significado que uma dada interrogação, numa base de dados. Com os dados das competições FireFAQ 2011 e FireFAQ 2012, obtivemos uma precisão na primeira posição da lista de resultados de 0.83 e 0.73, respetivamente.

5.1 Sumário das Contribuições

É reportado neste trabalho um conjunto de experiências que usam o conhecido Microsoft Research Paraphrase Corpus, no qual conseguimos uma classificação com uma exatidão de 0.77, e uma medida F1 de 0.84. Assim, é mostrado empiricamente que algoritmos de aprendizagem usando características relativamente simples podem obter resultados semelhantes aos resultados do estado-da-arte nesta tarefa, que tem uma importância cada vez maior.

Foi, também, efetuado um conjunto de experiências que usam os dados dos concursos FireFAQ 2011 e FireFAQ 2012, nos quais obtivemos uma classificação de precisão na primeira posição da

lista de resultados de 0.83 e 0.73, respetivamente. Mostrando empiricamente que os algoritmos usados para o Microsoft Research Paraphrase Corpus podem ser usados quer na deteção de relações de paráfrases entre duas frases como na detecção de uma paráfrase para com um conjunto de perguntas frequentes (FAQ).

5.2 Trabalho Futuro

Mesmo com os resultados interessantes, há ainda bastantes ideias para melhoramentos futuros. Por exemplo, gostaríamos de experimentar o uso de regras de transformação de frases usando padrões gramaticais diferentes (i.e. transformando voz passiva para voz activa). Gostaríamos também de introduzir outras características baseadas em métricas de avaliação das áreas de tradução automática, sumarização textual ou alinhamento sequencial em geral.

Para trabalho futuro, gostaríamos ainda de fazer experiências em diferentes conjuntos de dados (e.g. noutras línguas) e em diferentes domínios que também envolvessem o cálculo da relação semântica entre frases. Um dos domínios que tem que ver com a deteção de paráfrases é, por exemplo, a identificação de metáforas, como é apresentado em estudos anteriores por Shutova et al. (2012) e por Bollegala and Shutova (2013).

Bibliografia

W. Blacoe and M. Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning*, 2012.

D. Bollegala and E. Shutova. Metaphor Interpretation Using Paraphrases Extracted from the Web. In *PLoS ONE*, volume 8, 2013.

L. Breiman. Random forests. *Machine Learning*, 45(1), 2001.

C. Brockett and W. B. Dolan. Support Vector Machines for Paraphrase Identification and Corpus Construction. In *Proceedings of the 3rd International Workshop on Paraphrasing*, 2005.

W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, 2003.

D. Contractor, L. V. Subramaniam, P. Deepak, and A. Mittal. Text Retrieval Using SMS Queries: Datasets and Overview of FIRE 2011 Track on SMS-Based FAQ Retrieval. In *Proceedings of the Forum for Information Retrieval Evaluation*, 2013.

D. Das and N. A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and of the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, 2009.

L. R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3), 1945.

B. Dolan, C. Quirk, and C. Brockett. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal Machine Learning Research*, 9, 2008.

S. Fernando and M. Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium on Computational Linguistics in the UK*, 2008.

A. Finch, Y. H, and E. Sumita. Using Machine Translation Evaluation Techniques to Determine Sentence-Level Semantic Equivalence. In *Proceedings of the International Workshop on Paraphrasing 2005*, 2005.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 1997.

A. Islam and D. Inkpen. Second order co-occurrence PMI for determining the semantic similarity of words. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, 2006.

A. Islam and D. Inkpen. Semantic similarity of short texts. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, 2007.

P. Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37, 1901.

M. A. Jaro. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406), 1989.

Z. Kozareva and A. Montoyo. Paraphrase Identification on the Basis of Supervised Machine Learning Techniques. In *Proceedings of the 5th International Conference on Advances in Natural Language Processing*, 2006.

A. Lavie and S. Banerjee. Meteor: an automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the 2nd Workshop on Statistical Machine Translation*, 2005.

V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady.*, 10(8), 1966.

M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12), 2004.

N. Madnani, J. Tetreault, and M. Chodorow. Re-examining Machine Translation Metrics for Paraphrase Identification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2012.

P. Malakasiotis. Paraphrase Recognition Using Machine Learning to Combine Similarity Measures. In *Proceedings of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing 2009 Student Research Workshop*, 2009.

R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial intelligence*. AAAI Press, 2006.

M. K. Odell and R. C. Russell. Soundex Phonetic Comparison System. [U.S. Patents 1261167 (1918), and 1435663 (1922)].

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002.

L. Philips. Hanging on the Metaphone. *Computer Language Magazine*, 7(12), 1990.

L. Philips. The double metaphone search algorithm. *C/C++ Users Journal*, 18(6), 2000.

L. Qiu, M.-Y. Kan, and T.-S. Chua. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2006.

J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10), 2006.

L. Rokach. Ensemble-based classifiers. *Artificial Inteligence Review*, 33(1-2), 2010.

V. Rus, P. McCarthy, M. Lintean, D. McNamara, and A. Graesser. Paraphrase identification with lexico-syntactic graph subsumption. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*, 2008.

E. Shutova, T. Van de Cruys, and A. Korhonen. Unsupervised Metaphor Paraphrasing using a Vector Space Model. In *Proceedings of International Conference on Computational Linguistics*, 2012.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas*, 2006.

R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the Conference on Neural Information Processing Systems*, 2011.

J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

S. Wan, M. Dras, R. Dale, and C. Paris. Using dependency-based features to take the 'para-farce' out of paraphrase. In *Proceedings of the 4th Australasian Language Technology Workshop*, 2006.

Y. Zhang and J. Patrick. Paraphrase Identification by Text Canonicalization. In *In Proceedings of the 3rd Australasian Language Technology Workshop*, 2005.

S. Zhua, J. Wua, H. Xiongb, and G. Xiaa. Scaling up Top-K Cosine Similarity Search. 2009.

U.-Q. Zia and A. Wasif. Paraphrase Identification using Semantic Heuristic Features. *Research Journal of Applied Sciences, Engineering and Technology*, 4(22), 2012.