Uma *Benchmark* para *Kernels* de Extração de Relações

João L. M. Pereira, Gonçalo Simões, Helena Galhardas, and Bruno Martins

INESC-ID e Instituto Superior Técnico, Universidade de Lisboa
{joaoplmpereira,goncalo.simoes,helena.galhardas,
bruno.g.martins}@tecnico.ulisboa.pt

Resumo A extração de relações entre entidades em textos é uma tarefa importante no contexto de extração de informação. Esta tarefa envolve a identificação de relações entre duas entidades e a atribuição de um tipo a essa relação. A extração de relações é tipicamente abordada como um problema de classificação automática supervisionada entre pares de entidades. A tarefa de extração de relações envolve etapas de pré-processamento como a segmentação do texto, o reconhecimento de entidades, e a anotação a nível morfológico e sintático. Nos estudos anteriores nesta área, a forma como os dados são pré-processados varia entre as várias propostas, tornando injustas e inconclusivas as comparações das técnicas de classificação para extração de relações. Uma destas técnicas é a utilização de *kernels* que permitem a comparação de estruturas complexas. Neste artigo, a contribuição que propomos é uma *benchmark* ou estudo comparativo para a comparação de diferentes *kernels* para a tarefa de extração de relações. Especificamente, propomos a aplicação de um pré-processamento comum e a utilização de um algoritmo de aprendizagem *online* para treinar modelos de *Support Vector Machines* com *kernels* desenhados para a classificação de pares de entidades candidatos a relações. Reportamos ainda os resultados de uma comparação sistemática utilizando conjuntos de dados já conhecidos na área.

Keywords: Extração de Relações, *Benchmark*, *Support Vector Machines*, Aprendizagem *Online*.

1 Introdução

Existem cada vez mais documentos textuais disponíveis em formato digital. Estes documentos contêm informação valiosa que, se devidamente identificada e estruturada, pode ser utilizada por diversas aplicações (e.g., sistemas de resposta a perguntas). A extração de informação consiste em obter automaticamente dados estruturados a partir de documentos textuais. Esta atividade é tipicamente composta por várias tarefas tais como segmentação, extração de entidades, normalização, resolução de co-referências e extração de relações [12].

Este artigo foca-se na extração de relações. Por exemplo, dado o texto: *"The Taliban group Tehreek-e-Taliban Pakistan claimed the attack on the Karachi airport in southern Pakistan"*, um sistema de extração de relações deve ser capaz

de identificar a relação entre as entidades terrorist *"Tehreek-e-Taliban Pakistan"* e location *"Karachi"*, e classificar esta relação como sendo uma relação do tipo outrage. O resultado será o tuplo ⟨*"Tehreek-e-Taliban Pakistan"*, *"Karachi"*⟩ da relação com esquema outrage(terrorist, location).

Existem diferentes tipos de técnicas para a extração de relações. Estas técnicas geralmente dividem-se em dois grandes grupos: (i) baseadas em regras, que usam regras manualmente desenhadas por especialistas; e (ii) baseadas em aprendizagem automática, que extraem relações através da análise automática de padrões e/ou correlações nos dados [13]. Quaisquer destas técnicas dependem de anotações linguísticas e/ou léxico-sintáticas dos dados. Denominamos as tarefas que obtêm estas anotações de tarefas de pré-processamento.

Atualmente, a grande maioria das técnicas de extração de relações é baseada em aprendizagem automática supervisionada (i.e., técnicas que identificam padrões em dados previamente anotados, que constituem um conjunto de treino). Nestas técnicas, o conjunto de treino é constituído por exemplares de dados (*data instances*). Cada exemplar de dados é representado por um vetor de números reais. Uma posição no vetor corresponde a uma característica relevante (*feature*) do exemplar. Cada exemplar é acompanhado pela etiqueta da classe a que pertence indicada por um escalar. No contexto de extração de relações, um exemplar de dados corresponde a um par de entidades, e portanto as suas características relevantes (i.e., posição das palavras, dependências gramaticais) estão organizadas em estruturas complexas (e.g., sequências ou grafos de dependências). O mapeamento de estruturas complexas em posições de um vetor é uma tarefa difícil e que resulta em modelos pouco eficientes, devido ao elevado número de dimensões do espaço de características resultante. Uma solução para lidar com este problema consiste na utilização de funções *kernel* que, através da comparação implícita de estruturas complexas num espaço de características muito grande (ou potencialmente infinito), permitem obter modelos mais eficientes. A utilização destes *kernels* é uma prática comum na comunidade científica [13]. Os modelos do tipo *Support Vector Machines* (SVMs) constituem uma técnica de aprendizagem supervisionada que permite incorporar *kernels*.

Ao longo dos anos, tiveram lugar diversas competições (e.g., MUC [5], ACE [6] ou SemEval [8]) com o objetivo de comparar sistemas de extração de relações. No entanto, estas competições não permitem tirar conclusões justas sobre os *kernels* para extração de relações. Vários sistemas de extração de relações baseados em *kernels* foram avaliados com os conjuntos de dados destas competições. No entanto, estas avaliações utilizaram pré-processamentos e formas de treinar SVM diferentes, tornando a comparação do comportamento dos *kernels* injusta.

Inspirados pelo trabalho de Marrero et al. [11], no qual foi produzida uma *benchmark* para sistemas de extração de entidades, propomos, neste artigo, uma *benchmark* para comparar *kernels* propostos para a tarefa de extração de relações. Esta *benchmark* foi desenvolvida utilizando a *framework* para extração de relação, REEL [1][1], e é constituída por: (i) duas coleções de documentos de domínios distintos (AImed e SemEval); (ii) um pré-processamento comum

[1] http://reel.cs.columbia.edu/

para cada modelo SVM e constituído pelas seguintes fases: segmentação, normalização de palavras, normalização de maiúsculas, etiquetação gramatical, e análise de dependências; (iii) um conjunto de modelos SVM baseados em *kernels* do estado-da-arte; (iv) um algoritmo de aprendizagem *online* que treina os modelos SVM envolvendo os diferentes *kernels*; e (v) um processo de validação que utiliza métricas de convergência, de qualidade e de tempo de execução.

O resto do artigo encontra-se organizado do seguinte modo. A Seção 2 apresenta os conceitos fundamentais. A Seção 3 descreve a *benchmark* proposta. Na Seção 4, são detalhadas as configurações das experiências e é realizada uma análise exaustiva dos resultados. Finalmente, na Seção 5, terminamos o artigo com conclusões finais e ideias para trabalho futuro.

2 Conceitos Fundamentais

Nesta seção, apresentamos os conceitos fundamentais relacionados com a tarefa de extração de relações. Na Seção 2.1, introduzimos conceitos básicos de aprendizagem automática, em particular SVMs e aprendizagem *online*. Na Seção 2.2, descrevemos *kernels* para extração de relações.

2.1 SVMs e Aprendizagem *Online*

As técnicas mais comuns para realizar tarefas de extração de relações são as técnicas de aprendizagem automática supervisionada. Os SVMs [10] constituem uma das técnicas de aprendizagem automática supervisionada mais populares. Esta técnica produz um modelo de classificação binária, que distingue dados de duas classes através de um hiperplano que as separa. O maior desafio da utilização desta técnica é encontrar o hiperplano ótimo que separe os dados das duas classes. As representações dos hiperplanos é feita através do seu vetor normal $\mathbf{w}$. De forma a obter o vetor ótimo, $\mathbf{w}^*$, define-se uma função objetivo $f(S;\mathbf{w})$ que, através do conjunto de dados de treino S, identifica a proximidade de um vetor $\mathbf{w}$ ao vetor $\mathbf{w}^*$ para o problema. O conjunto de dados de treino é composto por exemplares de dados que, no contexto de extração de relações, são os pares de entidades candidatos a relação. Um exemplar do treino $(\mathbf{x}, y) \in S$ é constituído por um vetor $\mathbf{x}$ de entrada e um escalar $y \in \{-1, 1\}$ que representa a classe binária. Uma posição no vetor $\mathbf{x}$ corresponde a uma característica relevante do exemplar. A função objetivo é dada por $f(S;\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||^2 + \sum_{(\mathbf{x},y)\in S} loss(\mathbf{x}, y : \mathbf{w})$, e é composta por uma regularização $||\mathbf{w}||$ que é utilizada como uma medida da complexidade do modelo, e uma função de perda $loss(\mathbf{x}, y : \mathbf{w}) = \max(0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle)$ que avalia o modelo perante os dados de treino. O parâmetro λ é introduzido pelo utilizador por forma a determinar a importância da regularização.

Na tarefa de extração de relações, representar um exemplar através de um vetor é uma tarefa difícil que resulta em representações fracas dos dados. Uma alternativa mais satisfatória é representar os exemplares de dados por estruturas mais complexas e informativas (i.e., grafos ou sequências). De forma a comparar dois exemplares dos dados representados por estruturas complexas,

podem ser usados *kernels*. Os *kernels* devolvem um valor de similaridade entre os dois exemplares. Matematicamente, os modelos SVM podem ser treinados apenas com produtos internos entre vetores. Os *kernels* são um substituto para o produto interno entre vetores no espaço cartesiano.

O treino de um modelo SVM é efetuado através de algoritmos de otimização da função objetivo $f(S; \mathbf{w})$. Os algoritmos que tentam otimizar a função objetivo enquanto recebem os exemplares dos dados individualmente são denominados de algoritmos de aprendizagem *online*. Estes algoritmos são geralmente mais rápidos que outros tipos de algoritmo de otimização. O Pegasos [14] é um algoritmo de aprendizagem *online* que treina eficientemente modelos SVM.

Até este momento, apresentámos modelos SVMs enquanto classificadores binários. No entanto, grande parte dos problemas de classificação envolvem mais do que duas classes. Estes problemas são chamados de problemas multi-classe. Usando apenas um classificador SVM não é possível resolver diretamente um problema deste tipo. Para que isso seja possível são habitualmente usadas heurísticas, por exemplo a *One-VS-One* [9] e a *One-VS-All* [9]. Na Seção 3.3, fornecemos mais detalhes sobre o comportamento da heurística *One-VS-One* que usamos neste trabalho.

2.2 *Kernels* para Extração de Relações

Esta seção descreve *kernels* do estado-da-arte para extração de relações. Estes *kernels* utilizam estruturas complexas para representar melhor os exemplares de dados:

***Subsequences Kernel (SSK)*:** Bunescu e Mooney [4] desenvolveram um *kernel* baseado em subsequências. O *SSK* é composto pela soma de três *sub-kernels* que comparam subsequências esparsas de palavras (i.e., sequências de palavras não necessariamente contíguas) em localizações diferentes na frase: antes do par de entidades, depois do par de entidades e entre o par de entidades. Este *kernel* usa uma função que calcula a semelhança entre palavras através do número de características que as palavras têm em comum (e.g., a forma das palavras ou as suas classes gramaticais).

***Shortest Path Kernel (SPK)*:** Bunescu e Mooney [3] propuseram também um *kernel* baseado na comparação entre grafos de dependências. Um grafo de dependências é uma grafo dirigido que representa as dependências gramaticais entre as palavras de uma frase. Cada nó representa uma palavra e cada arco representa a dependência entre as duas palavras representadas pelos nós. O *SPK* tem a particularidade de usar apenas o sub-grafo que contém o caminho mais curto entre as entidades. Só os grafos com o mesmo número de nós no caminho mais curto são comparados (i.e., para os outros, o *kernel* devolve o valor zero). O valor final do *kernel* é calculado através da multiplicação dos valores da comparação dos nós e dos arcos. Os nós que existem na mesma posição dos sub-grafos são comparados primeiro entre eles usando a mesma função do *SSK*, que calcula a semelhança entre palavras através do número de características que as palavras têm em comum. Os arcos também são comparadas devolvendo 1 se têm o mesmo sentido e 0 caso contrário.

Tabela 1: Caracterização estatística dos conjuntos de dados.

	Frases	Pares de entidades candidatos	Entidades	Relações	Tipos de Relações
AImed	1159	5471	3754	996	1
SemEval	10717	10717	21434	8853	9

***Bag-Of-N-Grams Kernel (BNK)*:** Giuliano et al. [7] propuseram um *kernel* que combina dois *kernels* mais simples, nomeadamente (i) um *kernel* de contexto global que considera a informação relativa a toda a frase e (ii) um *kernel* de contexto local que considera apenas as palavras à volta das entidades. À semelhança do *SSK*, o *kernel* do contexto global é separado em três *sub-kernels* que avaliam a semelhança de três localizações da frase: antes, entre, e depois do par de entidades. Estes *sub-kernels* baseiam-se em n-gramas em vez de subsequências. N-gramas são conjuntos pequenos de palavras contíguas de tamanho n (normalmente $n = 3$). Os *sub-kernels* de cada localização comparam o número de n-gramas em comum entre duas frases, independentemente da ordem. O *kernel* de contexto local divide-se em dois *sub-kernels*: um avalia as palavras à volta da primeira entidade enquanto o outro avalia as palavras à volta da segunda entidade. Estes *sub-kernels* têm em consideração uma sequência de palavras centrada nas entidades e constituída por 6 palavras (i.e., as 3 palavras antes da entidade e as 3 palavras depois da entidade). Nestes *sub-kernels*, a semelhança entre duas palavras é obtida através da contagem das suas características comuns, de forma semelhante ao apresentado nos *kernels* anteriores.

3 *Benchmark*

Esta seção apresenta o processo de desenvolvimento da *benchmark* para *kernels* de extração de relações. O objetivo desta *benchmark* é disponibilizar um processo de avaliação imparcial que permita aferir a adequabilidade de cada *kernel* na tarefa de extração de relações e que não dependa de fatores externos (i.e., pré-processamento). Este processo de avaliação é dividido nas seguintes tarefas: (i) pré-processamento dos conjuntos de dados descritos na Seção 3.1 usando as técnicas disponíveis pelas ferramentas apresentadas na Seção 3.2; (ii) treino e execução de modelos SVM baseados em *kernels* através de técnicas descritas na Seção 3.3; e (iii) avaliação dos *kernels* com as métricas descritas na Seção 3.4.

3.1 Conjuntos de Dados

Esta seção descreve dois conjuntos de dados considerados para a *benchmark*, nomeadamente o AImed e o SemEval. Ambos os conjuntos de dados são compostos por documentos na língua inglesa. A Tabela 1 apresenta um resumo das estatísticas destes conjuntos de dados.

AImed[2]: é composto por 255 resumos de artigos do Medline, dos quais 200 referem interações entre proteínas e os restantes 25 não referem nenhuma interação.

[2] ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/interactions.tar.gz

No total, o AImed contém 5471 pares de proteínas, dos quais 996 correspondem a pares com interação e 4475 correspondem a pares sem interação. Um exemplo de uma frase deste conjunto de dados é *"Activated FGFR3 predominantly interacts with GRB2."* Neste exemplo, as entidades são *"FGFR3"* e *"GRB2"* e formam um par candidato no qual se verifica uma interação de proteínas.
SemEval (Semantic Evaluation) [8]**:** é uma série contínua de avaliações para sistemas computacionais de análise semântica. Na edição de 2010, esta competição propôs um desafio de classificação de relações semânticas entre pares de entidades. O conjunto de dados associado a esta tarefa é composto por 8000 frases de treino e 2717 frases de teste. Cada frase é claramente identificada no documento e contém apenas um par de candidatos. Esta tarefa inclui 9 tipos de relações assimétricas: *Member-Collection*, *Cause-Effect*, *Component-Whole*, *Instrument-Agency*, *Entity-Destination*, *Product-Producer*, *Message-Topic*, *Entity-Origin*, e *Content-Container*. Um exemplo de uma frase deste conjunto de dados é *"The most common audits were about waste and recycling"*, onde a relação *Message-Topic* é estabelecida entre as entidades *"audits"* e *"waste"*.

3.2 Ferramentas para o Pré-Processamento Linguístico

As técnicas de extração de relações utilizam geralmente vários tipos de anotações linguísticas e/ou léxico-sintáticas. Como tal, antes de proceder à extração de relações a partir de texto em língua natural, é necessário pré-processar o texto para obter essas anotações. Para o pré-processamento dos conjuntos de dados usámos as seguintes técnicas e ferramentas:

1. **Segmentação em frases:** Separa o texto em frases. O texto foi processado com a biblioteca *Apache OpenNLP*[3], que é uma ferramenta de processamento de língua natural baseada em aprendizagem automática. Para segmentar o texto, esta biblioteca utiliza um modelo supervisionado de máxima entropia [2] previamente treinado com dados em Inglês[4].
2. **Segmentação em palavras:** Identifica as palavras para cada frase, mantendo a ordem delas no texto. As frases foram processadas com a biblioteca *Apache OpenNLP*. Usámos um modelo de máxima entropia previamente treinado com dados em Inglês[5].
3. **Normalização das palavras:** Etiqueta as palavras com a sua raiz, de forma que palavras da mesma família tenham a mesma etiqueta (e.g., *claimed* e *claimes* têm a mesma raiz, *claim*). Com recurso ao Snowball[6] usámos o algoritmo de *Porter Stemming*, que encontra a raiz para cada palavra.
4. **Normalização de maiúsculas:** Etiqueta cada palavra com duas representações diferentes, transformando-as segundo dois padrões. O primeiro padrão substitui cada carácter por um de quatro símbolos específicos, dependendo se é uma maiúscula, uma minúscula, um número ou outro tipo

[3] http://opennlp.apache.org/
[4] http://opennlp.sourceforge.net/models-1.5/en-sent.bin
[5] http://opennlp.sourceforge.net/models-1.5/en-token.bin
[6] http://snowball.tartarus.org/

de carácter. O segundo padrão é semelhante ao primeiro, mas quando existem mais carateres do mesmo tipo, a sequência é substituída pelo respetivo símbolo associado ao tipo seguido do carácter +. Por exemplo, a normalização de maiúsculas da palavra *"Karachi"* resulta em *Ccccccc* e *Cc+*.

5. **Etiquetação Gramatical:** Produz etiquetas gramaticais (*Part-of-Speech*) e etiquetas gramaticais genéricas (*Generic Part-of-Speech*) para cada palavra. A etiqueta gramatical representa a classe gramatical (por exemplo *claimed* etiquetado com verb past tense) que essa palavra tem no texto, enquanto que a etiqueta gramatical genérica (por exemplo verb) representa uma classe gramatical de alto nível, tendo esta menos tipos de classes gramaticais. As classes gramaticais também foram obtidas através da biblioteca *Apache OpenNLP*. Usámos um modelo de máxima entropia em conjunto com um dicionário de etiquetas, novamente treinado com conjuntos de dados em Inglês[7].
6. **Análise de Dependências:** Produz uma árvore ou um grafo de dependências para cada frase. Utilizámos a biblioteca *Stanford CoreNLP*[8] que contém um analisador sintático para identificar dependências entre as palavras. As dependências são produzidas através do uso de gramáticas livres de contexto.

Todas estas técnicas de pré-processamento são utilizadas para produzir as estruturas comparadas pelos *kernels*. A exceção é o analisador de dependências, que apenas é necessário para o *SPK*. A normalização das palavras, a normalização de maiúsculas e a etiquetação gramatical são pré-processamentos opcionais sob a forma de características das palavras que melhoram a precisão dos *kernels*. No caso do SemEval, a segmentação em frases não foi necessária, porque as frases já se encontram devidamente identificadas no texto. A extração de entidades também não foi necessária, pois os conjuntos de dados utilizados já incluem as entidades anotadas.

3.3 Técnicas de Aprendizagem

Para treinar os classificadores SVM usámos um algoritmo de aprendizagem *online* denominado Pegasos [14]. Geralmente, um modelo SVM não fica devidamente treinado apenas com uma passagem sobre o conjunto de treino. Como tal, é necessário um processo iterativo que realize múltiplas passagens sobre os dados de treino.

Em problemas de extração de relações nos quais apenas se procura um tipo de relação (e.g., extração de interações de proteínas do AImed) é suficiente utilizar um classificador binário. No entanto, se tivermos o objetivo de extrair vários tipos de relações (e.g., extração de relações no SemEval), é necessário utilizar técnicas multi-classe. Para resolvermos a tarefa com recurso a classificadores binários utilizámos uma heurística *One-VS-One* de forma a combinar os resultados de múltiplos classificadores. Esta técnica recorre a tantos classificadores binários

[7] http://opennlp.sourceforge.net/models-1.5/en-pos-maxent.bin
[8] http://nlp.stanford.edu/software/corenlp.shtml

quantos pares de classes. Assim, treinámos cada classificador somente com os dados anotados das classes que avalia. Na fase de teste, um exemplar de dados foi avaliado por todos os classificadores. Cada classificador votou na classe que atribuiu ao exemplar. No fim, escolhemos a classe mais votada.

3.4 Métricas

Nesta seção apresentamos as métricas que usamos para avaliar os *kernels*:
Convergência: Utilizamos o valor da função objetivo de um modelo SVM perante os dados de treino. Por convergência, entende-se uma maior proximidade com o modelo ótimo resultante de otimizar a função objetivo. Desta forma, o valor da função objetivo a cada iteração é naturalmente um bom indicador da convergência do modelo.
Qualidade para classificação binária: As medidas normalmente usadas para avaliar sistemas de classificação são a precisão, a abrangência e a medida F_1. Definindo r como um tipo de relações, então a precisão é a fração de relações de tipo r corretamente extraídas, relativamente ao número total de relações extraídas como sendo do tipo r. A abrangência é a fração de relações de tipo r corretamente extraídas, relativamente ao número total de relações do tipo r presentes no conjunto de dados. Estas medidas são habitualmente contraditórias: ao se aumentar uma, diminui-se a outra. Para avaliar globalmente um sistema de classificação é necessário combinarmos estas medidas. Uma solução para combinar estas duas medidas é usando a medida F_1 que corresponde à média harmónica da precisão e abrangência.
Qualidade para classificação multi-classe: Para o caso de problemas de classificação multi-classe é vantajoso estender a precisão, a abrangência e a medida F_1 de forma a produzir uma medida final. Estas medidas finais podem ser obtidas através de médias dos valores de precisão e de abrangência, nomeadamente através de macro-médias e de micro-médias. As macro-médias calculam diretamente a média da precisão ou da abrangência do sistema para as diferentes classes. As micro-médias calculam as médias das relações de qualquer tipo r nas diferentes categorias (i.e., corretamente extraídas, número total de extrações e número total presente nos dados). Por fim usamos as fórmulas da precisão ou da abrangência com as médias das diferentes categorias.
Tempo de execução: Correspondem aos tempos de execução das diferentes tarefas nomeadamente o treino de um modelo e a previsão do tipo de classe para um par de entidades. As durações foram extraídas em nano segundos do tempo de CPU, sendo posteriormente convertidas para unidades de mais fácil comparação (i.e., segundos, minutos ou horas).

4 Resultados da Benchmark

Nesta seção apresentamos a configuração e os resultados das experiências realizadas usando a *benchmark* apresentada na Seção 3. Na Seção 4.1, descrevemos as configurações utilizadas nas experiências. Na Seção 4.2, apresentamos a análise

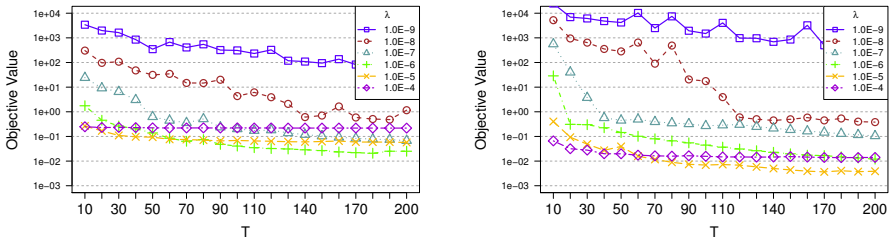
dos resultados de convergência dos modelos. Na Seção 4.3, analisamos os resultados da qualidade dos modelos, e na Seção 4.4, analisamos o tempo de execução de duas tarefas, nomeadamente o tempo de execução de treino do modelo e o tempo de execução médio da classificação de um exemplar.

4.1 Configurações

Nesta seção apresentamos as configurações das experiências que realizámos:
Conjuntos de dados: Usámos os conjuntos de dados descritos na Seção 3.1, nomeadamente o AImed e o SemEval. Ambos são pré-processados através das tarefas descritas na Seção 3.2. O AImed foi separado em 10 conjuntos de dados de treino e de teste (*folds*) sobre os quais utilizámos um processo de validação cruzada (*cross-validation*). O AImed contém apenas um tipo de relação, ou seja, extrair relações neste conjunto de dados é um problema de classificação binária. A competição SemEval disponibilizou uma separação em conjunto de dados de treino e de teste, pelo que neste conjunto de dados não foi necessário utilizar validação cruzada. Extrair relações neste conjunto de dados é um problema multi-classe, estando os dados anotados com 9 tipos de relações assimétricas, que no total, perfazem 19 classes distintas (duas para cada tipo mais uma para a não relação).
Técnicas de aprendizagem: Utilizámos as técnicas descritas na Seção 3.3, nomeadamente o algoritmo Pegasos para treinar os classificadores SVM; um classificador SVM por experiência para o conjunto de dados AImed; e a *One-VS-One* para o conjunto de dados SemEval, que utiliza 172 classificadores binários.
***Kernels*:** Utilizámos os *kernels* descritos na Seção 2.2, ou seja, o *SSK*, o *SPK* e o *BNK*. Estes *kernels* foram escolhidos por pertencerem ao estado-da-arte e por usarem representações distintas dos exemplares dos dados.
Parâmetros: O parâmetro λ, que regula a importância da regularização e da função de perda no treino do modelo SVM, tomou valores em $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$. O parâmetro do número de iterações, T, que é introduzido no algoritmo de aprendizagem *online* para indicar quantas passagens devem ser feitas sobre os exemplares de treino, tomou valores em: $\{50, 100, 150, 200\}$.
Software: A *benchmark* foi concretizada usando a *framework* para extração de relações REEL [1].
***Hardware*:** As experiências foram realizadas numa máquina com CPU Intel Core i5 M 460, 2.53 GHz e 4GB de memória RAM.

4.2 Convergência

Nesta seção, avaliamos o impacto dos parâmetros λ e T na aprendizagem dos modelos de extração de relações. Para o fazer, analisamos a variação da função objetivo do algoritmo de aprendizagem em função destes parâmetros. A Figura 1 apresenta um gráfico com esta variação. As nossas experiências mostraram que, para diferentes *kernels*, não existe uma diferença substancial nas nossas conclusões. Como tal, apenas apresentamos os resultados obtidos para o *BNK*.



AImed *Content-Container(e2,e1) vs Message-Topic(e1,e2)* no SemEval

Figura 1: Evolução dos valores da função objetivo do *BNK* para modelos de diferentes parâmetros de λ no AImed e SemEval.

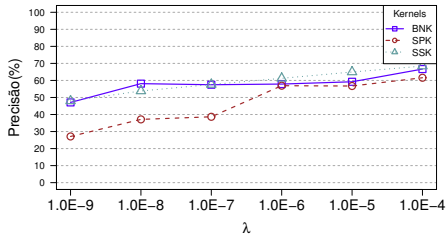
No geral, as iterações necessárias para que um modelo estabilize o valor obtido da função objetivo aumentam com a diminuição dos valores do parâmetro λ. De facto, para os menores valores que considerámos para λ não chegamos a atingir um modelo estável. Relativamente ao número de iterações, T, os valores da função objetivo começam a diminuir lentamente no gráfico depois da iteração número 100, e estabilizam a partir da iteração 150. No AImed os modelos que obtêm o valor mais baixo da função objetivo são os modelos nos quais o valor de λ é 10^{-6} e para o SemEval são os modelos nos quais o valor de λ é 10^{-5}. Esta situação ocorre porque, para conjuntos de dados com domínios de linguagem técnica como o AImed, os modelos tendem a ser mais complexos e por isso é mais difícil encontrar uma generalização adequada.

Dos resultados reportados nesta seção, podemos concluir que: (i) modelos com valores muito baixos do parâmetro λ obtêm piores resultados quando comparados com valores mais altos, e (ii) após 150 iterações, não existem praticamente diferenças significativas no valor da função objetivo. Nesta seção avaliámos o impacto dos parâmetros λ e T no treino dos modelos. Na próxima seção iremos avaliar os resultados dos modelos usando as métricas de qualidade.

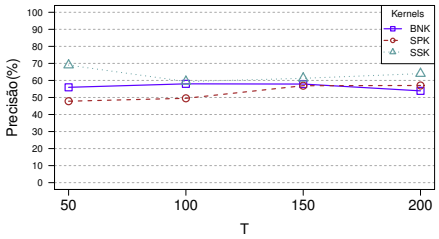
4.3 Qualidade dos Modelos SVM

Nesta seção, avaliamos os resultados de qualidade dos modelos SVM treinados nestas experiências. Para isso, aplicámos as métricas de qualidade que apresentámos na Seção 3.4. Começamos por avaliar o impacto da variação dos valores do parâmetro λ e do parâmetro T na qualidade dos modelos SVM sobre o AImed, e depois sobre o SemEval.

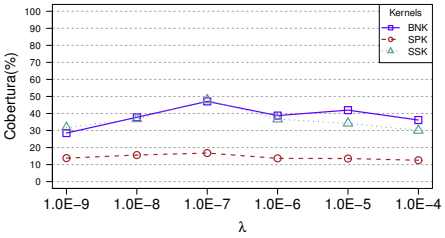
As Figuras 2 e 3 mostram resultados da análise de qualidade dos modelos SVM para o AImed e para o SemEval respetivamente. Em ambos os conjuntos de dados, o *BNK* e o *SSK* obtêm resultados semelhantes nas medidas F_1 (menos de 5% de diferença). Os valores de precisão do *SSK* são ligeiramente melhores que os do *BNK* (entre 1% e 5%). Para os valores de abrangência, verifica-se o oposto:



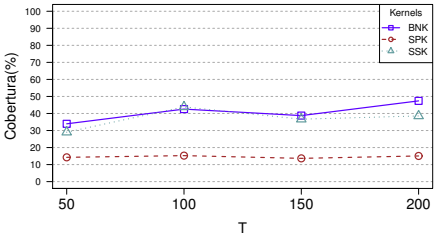
Precisão para $T = 150$



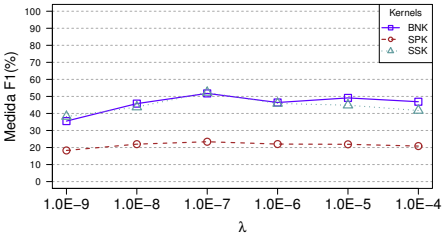
Precisão para $\lambda = 10^{-6}$



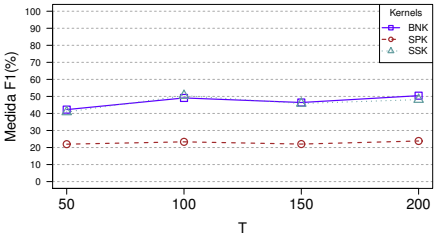
Abrangência para $T = 150$



Abrangência para $\lambda = 10^{-6}$



Medida F_1 para $T = 150$

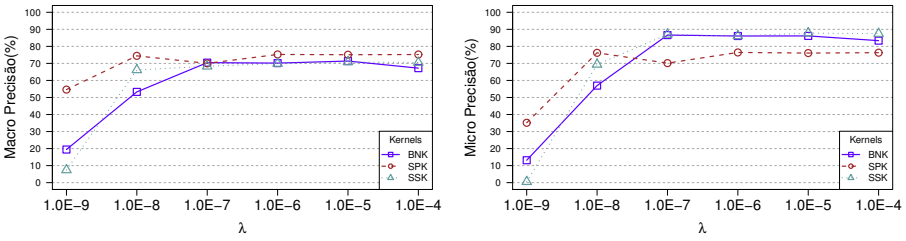


Medida F_1 para $\lambda = 10^{-6}$

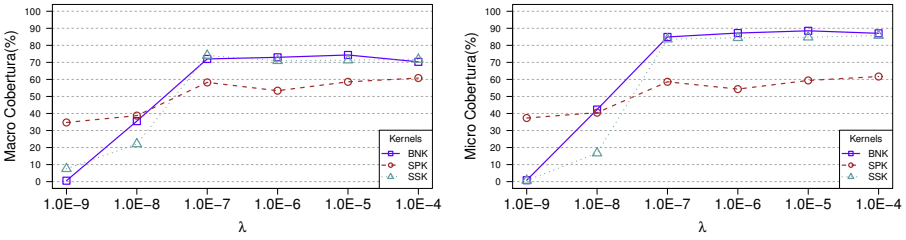
Figura 2: Avaliação sobre os dados de teste AImed.

o *BNK* obtém resultados sensivelmente melhores que o *SSK*. No geral, o *BNK* apresenta resultados mais equilibrados de precisão e abrangência que o *SSK*. De forma a explicar as diferenças entre estes dois *kernels* é necessário considerar as estruturas utilizadas. Ambos os *kernels* analisam as frases dividindo-as de forma semelhante. No entanto, no *SSK*, as características estão estruturadas em subsequências, que marcam uma ordem fixa nas palavras, enquanto que o *BNK* é mais flexível porque usa estruturas com conjuntos de n-gramas independentes da ordem em que estão na frase.

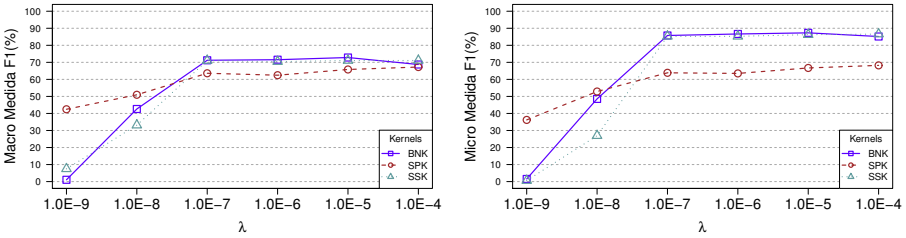
No geral, em ambos os conjuntos de dados, o *SPK* obtém resultados piores que os restantes *kernels* (e.g., entre 5% e 20% a menos). No AImed, esta diferença



Macro-média da precisão Micro-média da precisão



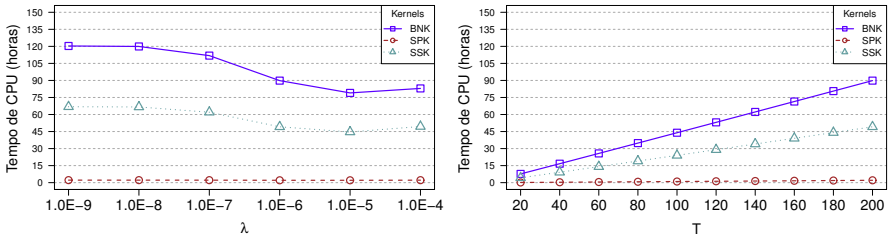
Macro-média da abrangência Micro-média da abrangência



Macro-média da medida F_1 Micro-média da medida F_1

Figura 3: Avaliação sobre os dados de teste SemEval em $T = 150$.

é mais significativa enquanto que no SemEval este *kernel* é apenas ligeiramente pior que os restantes. Uma vez que os valores de precisão obtidos pelo *SPK* são comparáveis aos dos outros dois *kernels*, as diferenças de qualidade deste *kernel* resultam da sua baixa abrangência. Para a macro-média da precisão no SemEval este *kernel* obtém os melhores resultados. Estas diferenças podem ser explicadas ao analisar a estrutura utilizada por este *kernel*. O *SPK* obtém bons resultados nas medidas de precisão porque, ao usar um caminho mais curto e ao comparar apenas caminhos com os mesmos tamanhos, este *kernel* torna-se mais inflexível na comparação. O *SPK* obtém ainda resultados de baixa qualidade (menos de 30%) no caso do AImed porque sendo este um conjunto de dados composto por



$T = 200$ $\lambda = 10^{-6}$

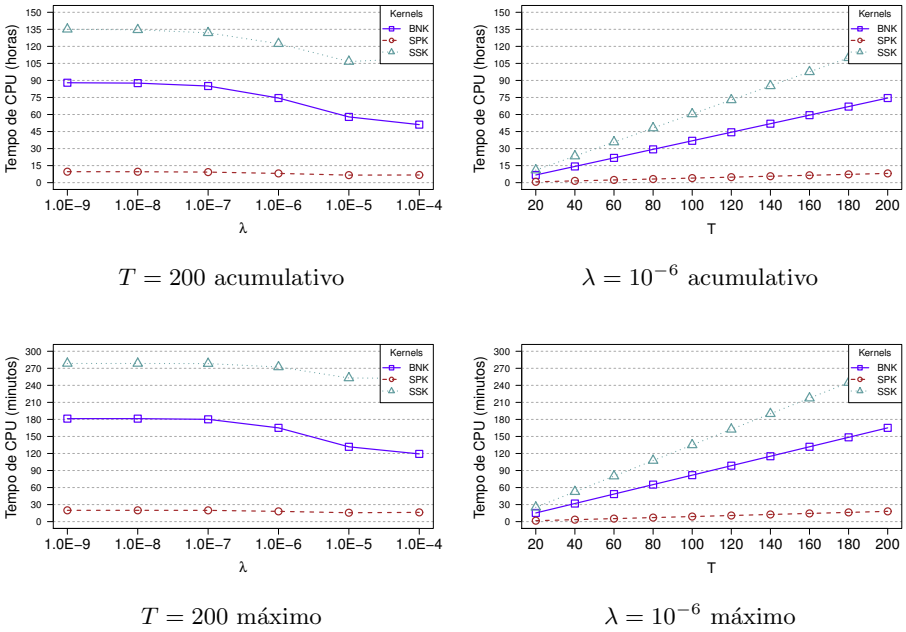
Figura 4: Tempo médio de treino para as 10 *folds* do AImed.

documentos técnicos de medicina, e tendo o processador de dependências sido treinado com noticias, as dependências usadas pelo *kernel* não são adequadas a este conjunto de dados. Devido aos erros do pré-processamento, consideramos injusta a comparação deste *kernel* no AImed. De forma a melhorar esta *benchmark* deve ser adicionado um processamento de dependências que venha a estar disponível para documentos técnicos de medicina.

Nesta seção podemos concluir que: (i) o *BNK* e o *SSK* têm um comportamento semelhante; (ii) o *BNK* é melhor na abrangência e mais equilibrado nas diferenças da precisão e abrangência; (iii) o *SPK* é o *kernel* que obtém, no geral, piores resultados, mas, no entanto, destaca-se do SemEval em termos de precisões. Para finalizar, comparando os melhores resultados em termos de macro F_1 de cada *kernel* com os sistemas avaliados no SemEval: o *BNK* com 73.56% obteria o sexto lugar, o *SSK* com 74.07% obteria o quinto lugar e o *SSK* com 67.65% obteria o décimo lugar. O sistema que conseguiu o primeiro lugar obteve 82.19% Embora a qualidade seja um fator muito importante na escolha do *kernel* para sistemas de extração de informação, o tempo de execução das tarefas também é um fator a ter em consideração. Na próxima seção, analisamos os *kernels* em termos do tempo de execução do treino e do teste.

4.4 Tempo de Execução

Nesta seção, comparamos os *kernels* em termos dos tempos de execução de treino e de teste. Os tempos de pré-processamento não foram contabilizados por dependerem muito do sistema usado. Para os tempos de execução de treino considerámos uma estimativa do tempo de CPU que foi calculado a partir do número de chamadas ao *kernel* a multiplicar pelo tempo médio de comparação de dois exemplares de dados. O tempo de CPU do algoritmo não foi considerado relevante nesta análise pois quando comparado com o tempo usado para calcular os *kernels* é significativamente baixo (i.e., menos de 1% do tempo total de execução). Enquanto no AImed o tempo de execução corresponde ao tempo de treino de um classificador, no SemEval consideramos a soma dos tempos de treino de todos os classificadores utilizados para o problema multi-classe.



$T = 200$ acumulativo

$\lambda = 10^{-6}$ acumulativo

$T = 200$ máximo

$\lambda = 10^{-6}$ máximo

Figura 5: Durações para o treino sobre o SemEval, valores acumulados e máximos.

Nas Figuras 4 e 5 reportamos os tempos de treino, respetivamente no AImed e no SemEval. Em ambos os conjuntos de dados, o *SPK* é significativamente mais rápido que os outros dois *kernels*. No entanto é de salientar que embora o *SPK* seja rápido de calcular, o seu pré-processamento é geralmente muito mais dispendioso que os pré-processamentos dos outros *kernels* por necessitar do processo de análise de dependências. Ao ter apenas em conta o caminho mais curto entre as entidades do grafo, este *kernel* é mais rápido porque: (i) compara menos características da frase que os outros dois *kernels*; (ii) ignora uma grande quantidade de comparações uma vez que só compara frases cujo número de nós no caminho mais curto entre as duas entidades seja igual.

No AImed, o *SSK* é mais rápido que o *BNK*. No entanto, no SemEval, esta diferença inverte-se. Esta diferença está relacionada com os tamanhos das frases em cada um dos conjuntos de dados (i.e., o tamanho médio de uma frase no AImed é significativamente maior que no SemEval). O *SSK* utiliza um algoritmo de programação dinâmica que torna a comparação rápida mesmo que o tamanho das frases aumente. O *BNK* é muito penalizado quando o tamanho da frase aumenta, especialmente quando há uma grande diversidade de palavras nas frases, o que é o caso no AImed.

Sabendo que a tarefa de extração de relações do SemEval é um problema multi-classe, é necessário treinar múltiplos classificadores para a realizar. Uma maneira de otimizar este processo é paralelizar o treino destes classificadores

(i.e., treinar cada classificador separadamente em diferentes *cores* de CPU). Assumindo uma paralelização perfeita do treino conseguimos treinar os modelos de forma significativamente mais rápida. Na prática é muito difícil conseguir paralelizações perfeitas uma vez que podemos não ter recursos suficientes para treinar um classificador por *core*. No entanto, sabemos que existe um número limitado de classificadores (aproximadamente 10 no SemEval) que se destacam no consumo do tempo de CPU (por terem um maior número de dados de treino). Assim, se conseguirmos paralelizar o processo em 10 tarefas (*threads*) que possam ser divididas por 10 *cores* conseguimos diminuir significativamente o tempo de CPU. As diferenças entre os *kernels* mantêm-se.

Em relação à variação do parâmetro λ, concluímos que existe uma relação entre os valores atribuídos ao λ e o tempo de execução do treino. Modelos com parâmetros λ menores têm um tempo de execução menor. Na análise da Seção 4.2, este valor estava associado à estabilidade e convergência dos modelos, em que nessa comparação, os valores mais baixos eram os que estabilizavam primeiro. Quando fixámos o parâmetro λ e variámos as iterações pudemos verificar que os tempos crescem linearmente em relação ao número de iterações.

Os tempos de teste são geralmente constantes para ambos os conjuntos de dados. Apenas a evolução do número de iterações no treino provoca algumas subidas mínimas nos tempos de teste. Assim, é suficiente analisar o tempo médio de classificação de cada exemplar. Com o AImed, são necessários 168ms para o *BNK*, 8ms para o *SPK*, e 290ms para o *SSK* para classificar um exemplar de teste. Já no SemEval são necessários 11ms para o *BNK*, 0.68ms para o *SPK*, e 6ms para o *SSK* para classificar cada exemplar. Sem surpresas, os resultados obtidos são semelhantes aos resultados da fase de treino analisados anteriormente.

Nesta seção, podemos concluir que: (i) o *SPK* é o mais rápido em treino e em teste. (ii) O *BNK* e o *SSK* têm durações nas mesmas ordens de grandeza. No entanto, (iii) o *SSK* é mais rápido que o *BNK* em conjuntos de dados em domínios técnicos, como o AImed, e que (iv) o *BNK* é mais rápido que o *SSK* para conjuntos de dados no domínio de notícias, como o SemEval.

5 Discussão e Trabalho Futuro

Neste trabalho, propomos uma *benchmark* para a comparação de *kernels* em tarefas de extração de relações baseadas em SVMs. Procedemos também a uma extensa análise experimental, utilizando a nossa *benchmark*, sobre três *kernels* do estado-da-arte, que nos permitiu tirar as seguintes conclusões: (i) modelos com valores mais baixos no parâmetro λ estabilizam mais rapidamente; (ii) tipicamente os modelos começam a estabilizar após o algoritmo de aprendizagem executar 100 iterações; (iii) o *BNK* e o *SSK* têm um bom desempenho em ambos os domínios; (iv) o *SPK* obtém resultados positivos apenas quando a análise de dependências é treinada em dados semelhantes; (v) o *SPK* é o *kernel* mais rápido de executar mas exige muito tempo de pré-processamento; (vi) existem poucas diferenças nos comportamentos do *BNK* e no do *SSK*; e (vii) o *BNK* tende a ser ligeiramente mais balanceado em precisão e abrangência.

Como trabalho futuro, pretendemos desenvolver técnicas que consigam combinar eficientemente os *kernels* para a extração de relações. Este método deve ser independente do domínio e deve poder ser utilizado com diferentes conjuntos de *kernels*. Para isso, devemos utilizar uma adaptação dos modelos SVM de forma a incluir na função objetivo os pesos atribuídos a cada *kernel*.

Agradecimentos. Este trabalho foi financiado pela Fundação para a Ciência e Tecnologia (FCT) através do financiamentos pluri-anual do INESC-ID correspondente ao projeto PEst-OE/EEI/LA0021/2013, através do projeto FCT DataStorm (ref. EXCL/EEI-ESS/0257/2012) e através da bolsa de doutoramento da FCT SFRH/BD/61393/2009.

Referências

1. P. Barrio, G. Simões, H. Galhardas, and L. Gravano. REEL: A Relation Extraction Learning Framework. In *IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, 2014.
2. A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.
3. R. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *HLT '05 Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, 2005.
4. R. Bunescu and R. J. Mooney. Subsequence kernels for relation extraction. In *9th Conference on Natural Language Learning (CoNLL)*, 2006.
5. N. A. Chinchor. Named entity task definition. In *7th Message Understanding Conference (MUC-7)*, 1998.
6. G. R. Doddington et al. The Automatic Content Extraction (ACE) program - tasks, data, and evaluation. In *4th International Conference on Language Resources and Evaluation (LREC)*, 2004.
7. C. Giuliano, A. Lavelli, and L. Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2006.
8. I. Hendrickx et al. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *5th International Workshop on Semantic Evaluation (SemEval)*, 2010.
9. C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.
10. T. Joachims. Text categorization with suport vector machines: learning with many relevant features. In *10th European Conference on Machine Learning (ECML)*, 1998.
11. M. Marrero, S. Sanchez-Cuadrado, J. M. Lara, and G. Andreadakis. Evaluation of named entity extraction systems. *Advances in Computational Linguistics. Research in Computing Science*, 41:47–58, 2009.
12. A. McCallum. Information Extraction: Distilling structured data from unstructured text. *ACM Queue*, 3:48–57, 2005.
13. S. Sarawagi. Information extraction. *Found. Trends Databases*, 1:261–377, 2008.
14. S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *24th International Conference on Machine learning (ICML)*, 2007.