5
Categorização de Textos

Neste capítulo são apresentados os fundamentos da área de Categorização de Textos, bem como um breve histórico da área. Além disso, uma pesquisa sobre os *softwares* de Mineração de Textos e suas possibilidades quanto à tarefa de Categorização.

5.1.
Introdução

Nos últimos anos, as tarefas de manipulação de documentos baseadas em conteúdo, conhecidas coletivamente como Recuperação de Informação, ganharam um status de destaque na área de Sistemas de Informação, devido ao aumento da disponibilidade de documentos em formato digital e a consequente necessidade de acessá-los e organizá-los de maneiras mais flexíveis.

Categorização de Textos (CT), a atividade de rotular textos em linguagem natural com categorias temáticas a partir de um conjunto pré-definido, é uma dessas tarefas (SEBASTIANI, 2002).

Na comunidade de pesquisa a abordagem dominante para este problema é baseada em técnicas de Aprendizado de Máquina: um processo indutivo cria um classificador que, a partir de um conjunto de documentos categorizados, aprende as características de cada categoria.

As vantagens desta abordagem sobre a heurística baseada em Engenharia do Conhecimento, que consiste na definição manual de um classificador - baseado em regras - por especialistas do domínio, são boa eficácia, economia considerável em termos de força de trabalho especializada e pouca dependência de domínio.

5.2.
Histórico da área de Categorização de Textos

A área de Categorização de Textos pode ser cronologicamente dividida em duas fases. Na primeira fase predominam as abordagens baseadas em Engenharia do Conhecimento e situou-se até o final da década 80. A fase seguinte foi marcada pelas abordagens ao problema baseadas em Aprendizado de Máquina e compreende a década de 90 aos dias atuais.

5.2.1.
1ª Fase - Até o final década de 80

CT remonta ao início dos anos 60 e até o final dos anos 80, a abordagem era baseada na Engenharia do Conhecimento, em que próprio especialista codifica o classificador através de regras que definem cada categoria. Nesse método é extremamente dependente de intervenção humana para a construção do classificador ou adaptação para outro domínio de conhecimento.

5.2.2.
2ª Fase - Década de 90 em diante

Nos anos 90, a abordagem mencionada acima começou a perder popularidade. O paradigma de Aprendizagem de Máquina, segundo o qual um processo indutivo cria um classificador que, a partir de um conjunto de documentos categorizados aprende as características de cada categoria, passou a centralizar os esforços acadêmicos.

As vantagens desta abordagem são: precisão comparável à obtida por especialistas humanos; e uma economia considerável em termos de força de trabalho especializada, uma vez que nenhuma intervenção de engenheiros do conhecimento ou especialista do domínio é necessária para a construção do classificador ou por sua portabilidade para um conjunto diferente de categorias ou domínio.

Atualmente, CT é, portanto, uma disciplina, interdisciplinar a Aprendizagem de Máquina e Recuperação de Informação, e, como tal, compartilha uma série de características com Mineração de Textos (Cavaleiro 1999; Pazienza 1997).

Ainda há um debate considerável sobre onde a fronteira exata entre essas disciplinas se encontra e a terminologia ainda está em processo de evolução. Mineração de Textos é cada vez mais utilizada para designar todas as tarefas que, através da análise de grandes quantidades de texto detecta padrões de uso e, extrai informações úteis. De acordo com este ponto de vista, CT é uma instância de Mineração de Textos.

Uma observação deve ser realizada sobre CT: muitas vezes, o termo **Categorização Automática de Textos**[22] (CAT) é utilizado para referenciar a abordagem de CT realizada por Aprendizagem de Máquina, mesmo que todas as etapas necessárias para a realização deste processo, como por exemplo, o ajuste dos parâmetros dos classificadores utilizados seja manual. Além disso, segundo (MANNING, RAGHAVAN, & SCHÜTZE, 2007), o termo também é utilizado para designar a descoberta de um conjunto de grupos que compartilham características comuns, como em (BORKO & BERNICK, 1963); uma tarefa normalmente chamada de Clusterização de Documentos.

5.3. Definição

De acordo com (LINDEN, 2008), Categorização de Textos é a atribuição de um valor booleano para cada par $(d_i, c_j) \rightarrow D \times C$, em que D é uma coleção de documentos e $C = \{c_j, \ldots, c_{|c|}\}$ um conjunto de categorias predefinidas. Partindo desse conceito inicial, a CT é formalmente descrita como a decisão de classificar $\phi d_i \notin c_j$, o valor-verdade $\mathbb{F}$ é associado ao par (d_i, c_j), caso contrário aplica-se o valor-verdade $\mathbb{V}$.

[22] Do termo inglês, Automated Text Categorization.

Para que o processo de decisão seja automatizado, é necessário que uma inferência seja realizada. Mais formalmente, a tarefa é aproximar a função de destino desconhecida $\breve{\phi} : D \times C \rightarrow \{\mathbb{V}, \mathbb{F}\}$ por meio de uma função $\phi : D \times C \rightarrow \{\mathbb{V}, \mathbb{F}\}$ denominada classificador, tal que $\breve{\phi}$ e ϕ coincidam tanto quanto possível.

Na realidade, a função $\breve{\phi}$ é uma aproximação da função ϕ. Essa função descreve o modo como os documentos da coleção D devem ser classificados em C. Os dados de entrada dessa função são os documentos ou suas características representativas e suas respectivas categorias. A saída é um conjunto de valores $\{\mathbb{V}, \mathbb{F}\}$.

(SEBASTIANI, 2002) também ressalta algumas premissas sobre o processo de Aprendizagem de Máquina em CT:

- As categorias são apenas rótulos simbólicos, e nenhum conhecimento adicional de seu significado está disponível.
- Nenhum tipo de conhecimento exógeno, ou seja, informações de qualquer tipo fornecidas para efeitos de classificação por uma fonte externa está disponível. A classificação deve ser baseada apenas no conhecimento endógeno, ou seja, o conhecimento extraído dos documentos. Metadados, tais como data de publicação, tipo de documento, fonte de publicação, etc., não estão disponíveis.

5.4. Tipos de Classificadores

Diferentes restrições podem ser aplicadas à tarefa de CT, dependendo do domínio do problema. Pode ser necessário que para um determinado número inteiro k, exatamente k (ou $\leq k$, ou $\geq k$) elementos de C sejam atribuídos a cada $d_j \in D$. O caso em que exatamente uma categoria deve ser atribuída a cada $d_j \in D$ é denominado monocategórico. O caso em que qualquer número de categorias de 0 à $|C|$ pode ser atribuído ao mesmo $d_j \in D$ é denominado multicategórico. Um caso especial do tipo monocategórico é o binário, em que cada $d_j \in D$ deve ser atribuído tanto à categoria c_i ou ao seu complemento $\overline{c_i}$.

Do um ponto de vista teórico, o caso binário é mais geral do que o multicategórico, uma vez que um algoritmo de classificação binária também pode ser usado para a classificação multicategórica: para isso, é necessário tratar o problema da classificação multicategórica de $\{c_1, \ldots, c_{|c|}\}$ em $|C|$ problemas independentes de classificação binária de $\{c_i, \overline{c_i}\}$, para $i = 1, \ldots, |C|$.

No entanto, essa abordagem requer que as categorias sejam estocasticamente independentes umas das outras, isto é, para qualquer c', c'', o valor de $\breve{\phi}(d_j, c')$ não depende do valor de $\breve{\phi}(d_j, c'')$, e vice versa. O inverso não é verdadeiro: um algoritmo de classificação multicategórico não pode ser usado para o caso binário. Na realidade, dado um documento d_j para ser classificado, o classificador multicategórico pode atribuir $k > 1$ categorias a d_j, e talvez não seja óbvio como escolher uma categoria mais adequada, ou o classificador multicategórico pode não atribuir a d_j categoria alguma, e talvez não seja óbvio como escolher a categoria menos inadequada de C.

5.5. Modelagem da categorização

Existem duas maneiras diferentes de se modelar um categorizador de textos. A primeira é denominada **categorização orientada a documento**[23]: dado $d_j \in D$, deseja-se encontrar todas as categorias $c_i \in C$ pertinentes a d_j. A alternativa é designada **categorização orientada a categoria**[24]: dada uma categoria $c_i \in C$, deseja-se encontrar todos os documentos d_j pertencentes a c_i.

Mais pragmática do que conceitual, esta distinção é importante quando os conjuntos C e D não estão inteiramente disponíveis desde o início. Também é relevante para a escolha do método de construção do classificador, pois alguns métodos permitem a construção de classificadores com uma inclinação definida para um ou outro modelo.

[23] Do termo inglês, *document-pivoted categorization.*

[24] Do termo inglês, *category-pivoted categorization.*

A categorização orientada a documento é, portanto, apropriada quando os documentos se tornam disponíveis em diferentes momentos no tempo, por exemplo, na filtragem de e-mail. Já a categorização orientada a categoria é mais adequada quando uma nova categoria $c_{|c|+1}$ pode ser adicionada a um conjunto existente $C = \{c_1, \ldots, c_{|c|}\}$ depois que uma série de documentos já tenham sido classificados em C, e esses documentos precisam ser reconsiderados para a classificação em $c_{|c|+1}$. A categorização orientada a documento é utilizada mais frequentemente do que PCC, pois a primeira situação é mais comum do que a última.

5.6. Tipos de categorização

Enquanto a automação completa da tarefa CT exige uma decisão $\mathbb{V}\, ou\, \mathbb{F}$ para cada par $(d_j,\ c_i)$, a automação parcial deste processo pode ter necessidades diferentes.

Por exemplo, dado $d_j \in D$ um sistema pode simplesmente classificar as categorias $C = \{c_1, \ldots, c_{|c|}\}$ de acordo com o grau de pertinência estimada à d_j, sem tomar qualquer decisão definitiva para qualquer uma das categorias. Tal lista ordenada seria de grande ajuda para um especialista humano encarregado de tomar a decisão final da categorização, uma vez que ele poderia, assim, restringir sua escolha às categorias do topo da lista, sem ter que examinar todo o conjunto.

Alternativamente, dado $c_i \in C$ um sistema pode simplesmente classificar os documentos em D de acordo o grau de pertinência estimada à c_i; simetricamente, para a classificação em c_i um especialista humano iria apenas examinar os documentos do topo da lista em vez de todo o conjunto de documentos.

Essas duas modalidades são denominadas categorização de textos ranqueada por categoria e categorização de textos ranqueada por documento (Yang, 1999), respectivamente, e são as contrapartidas evidentes de categorização orientada a documento e categorização orientada a categoria.

5.7.
Aplicações de Categorização de Textos

5.7.1.
Organização de documentos

Indexação com um vocabulário controlado é uma instância do problema geral da organização de base do documento. Em geral, muitas outras questões relativas à organização e armazenamento de documentos, seja para fins de organização pessoal ou estruturação de uma base corporativa de documentos, podem ser abordadas por técnicas de CT.

Por exemplo, nos escritórios de um jornal de classificados, os anúncios devem ser, antes de sua publicação, categorizados em categorias, tais como encontros, carros para venda, imóveis e outras. Jornais que lidam com um grande volume de anúncios classificados se beneficiariam de um sistema automático que define a categoria mais adequada para um determinado anúncio. Outras aplicações possíveis são a organização de patentes em categorias para tornar a sua busca mais fácil ou a identificação automática de artigos de jornal sob as seções apropriadas (Política, Esportes, Estilo de Vida, etc.).

5.7.2.
Filtragem de Documentos

Filtragem de documentos é a atividade de categorizar um fluxo de chegada de documentos enviados de forma assíncrona de um produtor de informação para um consumidor de informação (Belkin e Croft, 1992). Um caso típico é um *feed* de notícias, em que o produtor é uma agência de notícias e o consumidor é um jornal (Hayes et al., 1990). Neste caso, o sistema de filtragem deve bloquear o fornecimento de documentos em que provavelmente o consumidor não está interessado.

Pode ser vista como um caso de CT binária, isto é, a classificação dos documentos que entram em duas categorias disjuntas: relevantes e irrelevantes. Além disso, um sistema de filtragem também podem ainda classificar os

documentos considerados relevantes para o consumidor de informação em categorias temáticas. No exemplo acima, todos os artigos sobre esportes devem ser classificados de acordo com o esporte que lidam, de modo a permitir que os jornalistas especializados em esportes individuais acessem apenas os documentos de interesse potencial para eles.

5.7.3. Desambiguação Lexical de Sentido

A ambiguidade lexical é causada, fundamentalmente, pela existência de algumas relações semânticas interlexicais, principalmente a polissemia e a homonímia. De acordo com (Lyons, 1977), na polissemia uma mesma palavra tem dois ou mais significados diferentes, mas relacionados entre si, sendo que, normalmente, somente um dos significados se ajusta a um determinado contexto. Na homonímia duas ou mais palavras com significados totalmente distintos, sem traços comuns, são idênticas quanto ao som (homofonia) e/ou à grafia (homografia).

O problema da ambiguidade lexical pode, ainda, ser classificado como ambiguidade categorial ou ambiguidade de sentido (Ullmann, 1964).

A ambiguidade categorial ocorre quando as duas ou mais opções de significados de uma dada palavra são de diferentes categorias gramaticais. Na tradução, um exemplo de ambiguidade categorial causada pela relação de homonímia é a palavra do inglês *field*, que pode ser traduzida para as palavras "campo" (substantivo) ou "interceptar" (verbo), no português. Já um exemplo de ambiguidade categorial derivada da relação de polissemia é a palavra do inglês *eats*, que pode ser traduzida no português como "mantimentos, víveres, gêneros alimentícios" (substantivos) ou "come" (verbo "comer" conjugado na terceira pessoa singular, presente do indicativo).

A ambiguidade de sentido, por sua vez, ocorre quando as duas ou mais opções de sentido (ou tradução) de uma dada palavra têm a mesma categoria gramatical. Alguns exemplos são a palavra *know*, que pode ser traduzida como "saber" ou "conhecer", como um caso de polissemia, e a palavra light, que pode ser traduzida como "leve" ou "luz", como um caso de homonímia.

A ambiguidade categorial é, em geral, muito mais simples que a de sentido, uma vez que pode ser resolvida, na maioria das vezes, pela análise das características sintáticas das palavras, realizada por procedimentos de etiquetação gramatical (ver item 3.2.3.2) ou análise sintática (ver item 3.2.3.6), por exemplo. Procedimentos dessa natureza alcançam, atualmente, resultados bastante satisfatórios. A **resolução da ambiguidade de sentido**[25], por sua vez, exige a análise da semântica das palavras e, eventualmente, a análise do uso de tais palavras (realizadas por procedimentos de análise semântica e pragmática, por exemplo).

WSD é muito importante para muitas aplicações, como Processamento de Linguagem Natural e indexação de documentos pelo valor semântico dos termos. Pode ser vista como uma tarefa de CT, uma vez que contextos de ocorrência do termo sejam tratados como documentos e os sentidos do termo como categorias. Em geral, é abordado como um problema de categorização binária e modelo como categorização orientada a documento.

5.8. Aprendizagem de Máquina em CT

5.8.1. Aprendizagem Supervisionada

Nesta abordagem de Aprendizagem de Máquina, um processo indutivo cria automaticamente um classificador para uma categoria c_i observando as características representativas de um conjunto de documentos categorizados manualmente sob c_i ou $\bar{c}_i$ por um especialista no domínio. A partir destas características, o processo indutivo compila as características que um novo documento deve ter para ser classificado sob c_i.

Na terminologia de Aprendizagem de Máquina, o problema de classificação é uma atividade de aprendizagem supervisionada, uma vez que esse processo de

[25] Do ter inglês, word sense disambiguation (WSD)

aprendizagem é guiado pelo conhecimento das categorias e dos exemplos de treinamento que lhes são pertinentes.

5.8.2. Treinamento e Teste

Existem diversas estratégias para a execução do treinamento e, posteriormente, aplicação de testes. O treinamento tem como objetivo apresentar ao classificador exemplos que o farão conhecer e aprender sobre a massa textual. A aplicação de testes possibilita a avaliação do desempenho. A seguir, a descrição das principais estratégias relatadas na literatura:

- **Holdout**: Consiste em separar do conjunto de treinamento uma determinada porção, compondo o conjunto de teste. Usualmente, o teste utiliza 1/3 do conjunto total, mantendo o restante para treinamento. Apesar de simples e rápida de se aplicar, recebe críticas por não usar de forma otimizada o conjunto total de amostras – o classificador poderia ficar mais bem construído se utilizado todo o conjunto de treinamento – e pela própria aleatoriedade dos dados, isto é, o conjunto de teste pode acabar ficando "favorecido", levando a uma falsa conclusão da real adequação do treinamento.
- **K-Fold Cross Validation** (Validação Cruzada): Validação Cruzada é a metodologia de treinamento e teste que trabalha com o conceito de *folds*. Desta forma, o conjunto de amostras inicial é dividido em *k* subamostras. Destas *k* subamostras, uma subamostra é retida para ser utilizada na validação do modelo (conjunto de teste) e as *k-1* subamostras compõem o conjunto de treinamento. O processo é então repetido *k* vezes, de modo que cada uma das *k* subamostras seja utilizado ao menos uma vez como teste. O resultado final é a média do desempenho do classificador nas *k* iterações. O objetivo desta estratégia é aumentar a confiabilidade da avaliação, com o ônus de se despender mais tempo que a técnica anterior. Vale ressaltar que nada impede que as duas estratégias possam ser combinadas, com a aplicação da técnica de *holdout* como mais uma

forma de validar os resultados conseguidos com a Validação Cruzada, com o ônus de se despender muito mais tempo para a execução dos ciclos e de ser necessário mais dados para que os conjuntos (treinamento e teste) formados possam ter tamanho significante.

5.8.3. *k*-Nearest Neighbors

O *k-Nearest Neighbors* (KNN) é um algoritmo de aprendizagem supervisionado, pertencente a um grupo de técnicas denominado de *Instance-based Learning* ou *Lazy Learning* que tem sido usado em muitas aplicações no campo da Mineração de Dados, reconhecimento de padrões estatísticos, processamento de imagens e entre outros. É considerado um dos melhores métodos para a classificação de texto, é simples, efetivo e escalonável para grandes aplicações. Algumas aplicações de sucesso incluem reconhecimento de escrita à mão e imagens de satélite.

O algoritmo *k-Nearest Neighbors* classifica um dado elemento desconhecido (de teste) baseado nas categorias dos *k* elementos vizinhos mais próximos, ou seja, os elementos do corpus de treinamento que obtiveram os graus de similaridade mais altos com o elemento de teste, conforme ilustrado na Figura 24.

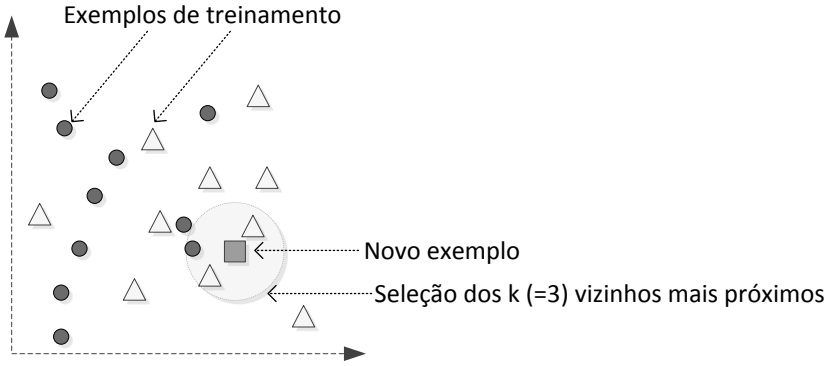

Figura 24 – Algoritmo KNN - Seleção baseada nos *k* (= *3)* vizinhos

Calcula-se a similaridade de cada um dos elementos do corpus de treino com o elemento que se quer classificar e então ordena os componentes da base de treino do mais similar ao menos. Dos elementos ordenados, selecionam-se os *k* primeiros, que irão servir como parâmetro para a regra de classificação.

Dois pontos importantes do kNN são: a regra de classificação e a função que calcula a similaridade. A regra de classificação diz como o algoritmo vai tratar a importância de cada uns dos *k* elementos mais próximos. A função de similaridade vai mensurar quão dois elementos são semelhantes de forma a poder identificar quais são os kNN.

Para $k = 1$ não existe regra de classificação, pois o elemento de classe desconhecida terá a mesma classificação do vizinho mais próximo. Para $k > 1$ é preciso uma regra para decidir a qual classe se atribuirá o elemento. Em (WANG, 2006) são revistas duas regras de classificação bem comuns: maioria na votação (*majority voting scheme*) e a soma do peso da similaridade (*weighted-sum voting scheme*). Na primeira, cada elemento tem uma influência igual, a classe escolhida será aquela que tiver mais representantes entre os *k* elementos. Na segunda, entre os *k* elementos, são somadas as similaridades dos elementos de mesma categoria, o elemento desconhecido será classificado na categoria que obtiver maior valor. As funções do cálculo de similaridade mais conhecidas na literatura são: a do cálculo do cosseno de dois vetores e a distância euclidiana.

5.8.4. SVM

Support Vector Machines ou Máquinas de Vetores de Suporte constituem uma técnica de Aprendizagem de Máquina supervisionada, utilizada para classificação e regressão. Em problemas de classificação, a ideia principal de uma máquina de vetores de suporte é construir um hiperplano como superfície de decisão de tal forma que a margem de separação entre exemplos positivos e negativos seja máxima, conforme ilustrado na Figura 25.

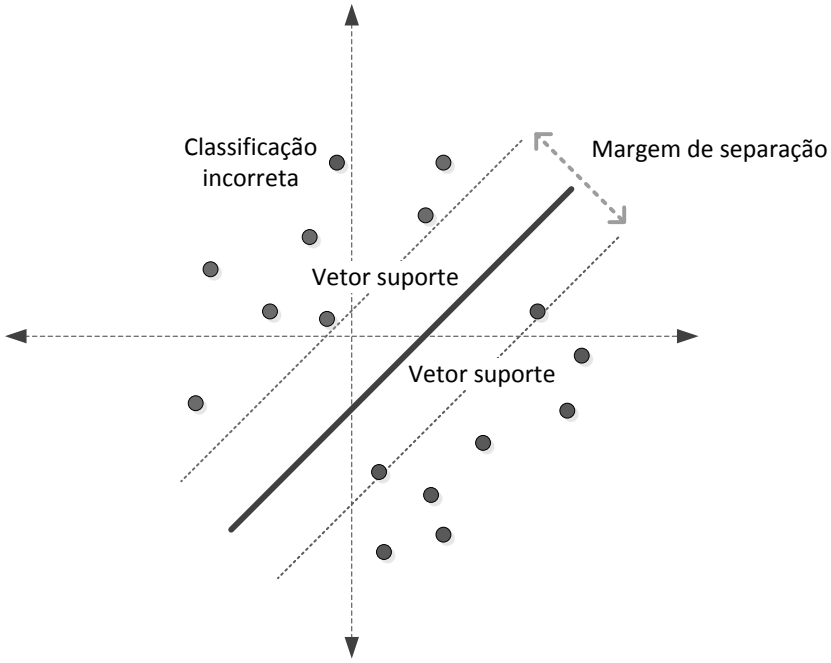

Figura 25 – Máquina de Vetores de Suporte

Aplicada com sucesso em diversas áreas, SVM tem sido muito utilizada em PLN, e com excelentes resultados em tarefas como Categorização de Textos (JOACHIMS, 1998) e Desambiguação Lexical de Sentido (LEE, NG, & CHIA, 2004).

SVM é um classificador binário. Para estender a utilização de SVMs em problemas de *k* classes (*k > 2*), há uma abordagem básica, conforme ilustrado na Figura 26, que é reduzir o problema de *k* classes a um conjunto de problemas binários. Para isto, duas abordagens são possíveis, e seguem os seguintes passos:

- Decomposição um por classe:
  1. Construção de *k* SVMs binárias para separar uma classe de todas as outras.
  2. Classificação nas *k* classes: o ponto x é classe da SVM com maior saída.
- Separação das classes duas a duas:
  1. Construção da SVM_{ij} para distinguir a classe *i* da classe *j*, $i \neq j$ e $i, j \in \{1,...,k\}$.
  2. Classificação na classe *i* ou *j* (voto para essa classe).
  3. Classificação final através de votação.

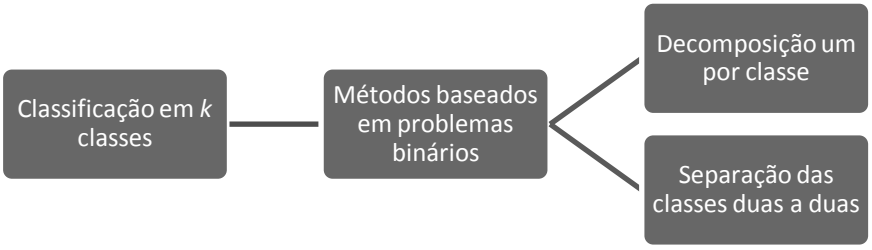

Figura 26 – Abordagens SVM para problemas não binários

Para ambas as abordagens vistas acima, podem ocorrer situações em que o resultado final não pode ser concluído sobre a classe a ser predita. Por exemplo, ao reduzir um problema de classificação com três classes distintas, é necessária a construção de três SVMs, sejam estas para a decomposição um por classe (SVM_1, SVM_2, SVM_3) ou separação das classes duas a duas (SVM_{12}, SVM_{13}, SVM_{23}). E nestes casos, pode ocorrer empate no resultado final, conforme ilustrado Tabela 9:

Tabela 9 - Resultados conflitantes de SVMs binárias

Um por classe	Saída SVM
SVM_1	Sim
SVM_2	Não
SVM_3	Sim

Duas a duas	Saída SVM
SVM_{12}	1 (sim)
SVM_{13}	3 (não)
SVM_{23}	2 (sim)

Outro detalhe importante sobre a utilização de SVMs em problemas de classificação é relativo ao valor do parâmetro *C*. que possui as seguintes características:

- Controla o compromisso entre a complexidade da maquina e o número de pontos não separáveis;
- Pode ser visto como uma forma de parâmetro de regularização e deve ser selecionado pelo usuário.

Em geral, são utilizadas heurísticas gulosas para o ajuste deste parâmetro, ou seja, inicia-se com valores pequenos e gradativamente aumenta-se o valor de *C*.

Importante também é definir o tipo de função *kernel* a ser utilizada pela SVM. A saber, são três funções disponíveis: Linear; Polinomial e RBF.

5.8.5. Combinação de Classificadores

A combinação de classificadores, uma alternativa para melhorar o processo de CT, é baseada na seguinte premissa: dada uma tarefa que exige o conhecimento de especialista, uma quantidade n de especialistas irá apresentar um resultado de melhor qualidade do que apenas um especialista apresentaria. Na Categorização Automática de Textos, esse esforço consiste em executar um conjunto de *k* classificadores $\{\phi_1, \ldots, \phi_k\}$ para a mesma tarefa de categorização e combiná-los de forma apropriada. Essa combinação de classificadores é caracterizada por uma escolha de *k* classificadores e de uma função combinatória.

A *performance* de um classificador pode ser ajustada para obter a melhor acurácia possível para uma determinada situação, mas os ajustes são uma tarefa complexa e, ainda, podem existir padrões em que mesmo o melhor classificador apresente falhas na categorização. Segundo (ALPAYDIN, 2004), para obter resultados produtivos na combinação de classificadores, as decisões de categorização tomadas pelos classificadores não podem ser as mesmas. Decisões iguais levam a resultados iguais; se for tomada a decisão errada, todos os classificadores erram. Decisões diferentes permitem a um classificador errar enquanto outros classificadores acertam, resultando em uma decisão final correta.

A principal discussão desse método é em relação à função combinatória. Dentre inúmeras funções combinatórias disponíveis, a votação, a seleção dinâmica de classificadores e a combinação adaptativa de classificadores são três funções combinatórias comumente utilizadas. Segue uma breve exposição sobre cada uma, a partir dos pressupostos apresentados por (SEBASTIANI, 2002) e (BENNETT, DUMAIS, & HORVITZ, 2005):

- A função combinatória mais simples é a escolha por voto majoritário. Nela a decisão final é obtida escolhendo-se a categoria com maior número de votos dentre os *k*

classificadores. Outra forma de aplicar a votação é atribuir pesos aos classificadores (votação com pesos); nesse caso o voto de cada classificador é relativo ao seu peso, para a decisão final.

- Um exemplo de seleção dinâmica de classificadores é a utilização do classificador mais eficiente para uma determinada categoria ou categorias. Esta função também pode ser denominada *Best By Class*.
- A combinação adaptativa de classificadores é descrita como uma função intermediária entre a votação com pesos e a seleção dinâmica de classificadores. Essa função agrega todos os votos dos classificadores, atribuindo pesos para a contribuição da decisão final conforme a eficiência de cada classificador em um corpus de avaliação.

Com base nessas funções combinatórias iniciais, diferentes autores descrevem suas próprias propostas de métodos combinatórios que, de uma forma ou outra, fazem uso dos conceitos expostos por (SEBASTIANI, 2002). A seguir são apresentados alguns exemplos de combinação de classificadores:

- O uso de *bagging* em (ALPAYDIN, 2004) descreve um método de votação onde os classificadores são treinados com pequenas alterações no corpus. Nesse caso, os classificadores devem ser treinados usando uma amostra aleatória do corpus, diferente a cada iteração. Para garantir tamanhos iguais de amostra para todos os classificadores, os documentos usados em uma iteração são repostos nas próximas iterações. Existe a possibilidade dos documentos serem usados mais de uma vez ou, até mesmo, nenhuma vez. O fator aleatório permite amostras, ao mesmo tempo, semelhantes e diferentes. Semelhantes porque podem compartilhar os mesmos documentos, e diferentes porque documentos diferentes são incorporados à

amostra. A principal desvantagem desse método é o fato de que o desempenho dos classificadores é baseado na probabilidade da escolha das amostras.

- O método de *boosting* (SEBASTIANI, 2002) apresenta um conceito intuitivo de aprimoramento, que se encaixa na definição de seleção dinâmica de classificadores. A ideia do *boosting* é aplicar um conjunto de *n* classificadores iterativamente sobre um corpus de treino. A cada iteração um novo classificador prioriza a categorização nos documentos onde o classificador anterior obteve a maior taxa de categorizações incorretas. Assim, o treinamento de novos classificadores não é baseado em probabilidade, como ocorre no método *bagging*. A desvantagem desse método é a exigência de um corpus suficientemente grande para o treinamento dos classificadores.
- O algoritmo *AdaBoost* (FREUND & SCHAPIRE, 1999), frequentemente citado na literatura, utiliza uma medida de peso para referenciar os documentos incorretamente categorizados, que recebem pesos maiores, enquanto documentos corretamente classificados recebem pesos menores. Utilizando a reposição de documentos, o algoritmo permite a escolha por documentos com pesos menores em corpora pequenos.
- (ALPAYDIN, 2004) exemplifica uma forma de combinação em cascata entre os classificadores. A ideia é ter *k* classificadores utilizados em sequencia, de acordo com sua complexidade ou custo de representação. Assim, os classificadores são aplicados a partir do mais simples ao mais complexo (FREUND & SCHAPIRE, 1999). Cada classificador garante um grau de confiabilidade em

sua decisão, para que os classificadores seguintes concentrem o esforço em categorizar os documentos com baixo índice de confiabilidade na categorização. Esse é um método muito semelhante ao *boosting*, a nova característica é o uso de classificadores diferentes no processo de categorização.

5.9. Ferramentas de Mineração de Textos

A seguir são descritas algumas destas ferramentas com suas características e aplicabilidade.

5.9.1. Weka

O software WEKA, desenvolvido por (HALL, FRANK, HOLMES, FAHRINGER, REUTEMANN, & WITTEN, 2009), é uma coleção de algoritmos de aprendizado de máquina para tarefas de Mineração de Dados. Os algoritmos podem ser aplicados diretamente a um conjunto de dados por meio de sua interface gráfica ou executados em outros aplicativos customizados via código Java. Contém ferramentas para pré-processamento de dados, classificação, regressão, clusterização, regras de associação e visualização. Também é bem adequado para o desenvolvimento de novos sistemas de aprendizagem de máquina.

Embora desenvolvido para lidar com informações estruturadas, o software dispõe de um filtro chamado *StringToWordVector* que realiza o processo de tokenização e remoção de *stopwords* de documentos textuais. Além disso, possui a habilidade de realizar o cálculo de relevância dos termos tokenizados segundo métricas como IDF e TF/IDF. Possui a opção de realizar a operação e *Case*

Folding que consiste em converter todos os termos para caixa baixa. Uma frequência mínima por termo pode ser definida fazendo com que este seja descartado pelo processo de tokenização. A figura exibe a tela de configuração desse filtro.

Após a aplicação desse filtro, obtém-se uma representação estruturada dos documentos, tornando possível a aplicação dos algoritmos tradicionais de Mineração de Dados disponíveis na ferramenta.

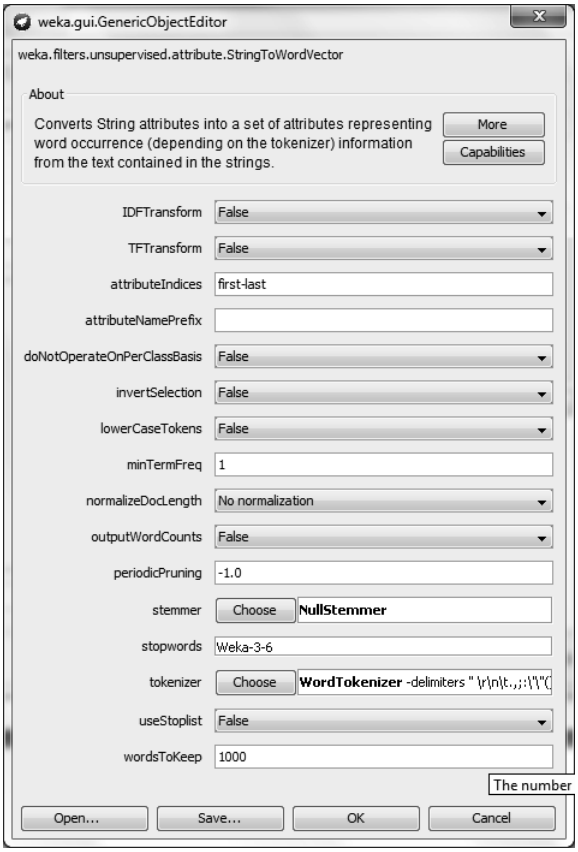

Figura 27 – Parâmetros de configuração do filtro *StringToWordVector* do software Weka

5.9.2. Text Mine

Text Mine, desenvolvido por (KONCHADY, 2006), é um conjunto de ferramentas de Mineração de Textos escritas em Perl. Necessita de um servidor *web* Apache e banco de dados MySQL.

Text Mine disponibiliza as seguintes funcionalidades:

- Coleta: a busca pode ser feita a partir de URLs, onde o *crawler* visita as páginas, construindo o conjunto de informações necessárias para as outras funcionalidades, ou em documentos locais à máquina.
- Extração de informação: é capaz de identificar entidades no texto como nomes, locais e organizações. Para isso, o usuário deve customizar um dicionário de entidades que serão identificadas.
- Clusterização: retorna uma coleção de grupos, onde cada grupo é composto por um número variável de documentos similares. Gera uma matriz de similaridades da coleção de documentos e utiliza algoritmo genético para agrupá-los segundo o grau de similaridade existente entre eles.
- Categorização: consiste em organizar as mensagens de texto em categorias.
- Sumarização: consiste em resumir artigos ou documentos da Web, extraindo palavras- chave que determinam o significado do documento.

5.9.3. TMSK

TMSK ou Text-Miner Software Kit é um pacote de softwares para mineração preditiva de textos. Possui funcionalidades para pré-processar documentos de textos em formato XML e disponibiliza implementações para as seguintes tarefas:

- Pré-processamento: implementa funcionalidades de tokenização, *stemming*, criação de dicionário e detecção de início e fim de sentenças.
- Classificação: possui classificadores baseados em regras de decisão, predição usando Naive Bayes e modelos de ranqueamento linear.
- Recuperação de informação: implementa o conceito de índices invertidos para busca de informação.
- Clusterização: realiza o agrupamento de documentos usando o algoritmo *k-means*.

- Extração de informações: identificação de entidades nomeadas.

5.9.4. RIKTEXT

O RIKTEXT é um pacote de softwares completo para categorização de documentos baseado em regras de decisão.

Seu objetivo é determinar o melhor conjunto de regras para a predição e classificação, onde o melhor conjunto é formado pelo menor número de regras com erro mínimo.

Os dados para este classificador devem estar na forma de tabela, onde cada linha corresponde a um documento e cada coluna corresponde a um termo do dicionário. Cada célula da planilha recebe valores booleanos indicando a presença ou a ausência da palavra no documento, ou a frequência linear do termo.

O RIKTEXT complementa o TMSK, disponibilizando métodos para construção e uso de regras para classificação de documentos. O formato dos dados de entrada do RIKTEXT é idêntico ao dos métodos de classificação apresentados no TMSK.

5.9.5. STATISCA Text Miner

O STATISTICA Text Miner é uma extensão opcional do STATISTICA Data Miner que compreende as etapas de coleta, pré-processamento e mineração do processo de *KDT*.

Dispõe de algoritmos para a redução de palavras ao seu próprio radical (lematização) e eliminação de *stopwords*. Assim, o aplicativo elimina uma parte insignificante do texto a ser analisado, reduzindo o número de termos da matriz de palavras de texto. Uma vez definidas as palavras com real valor para a análise, são aplicados algoritmos de mineração, por meio dos quais se derivam modelos preditivos para explicar a possibilidade de ocorrência de termos significantes em outros documentos.

As características principais do STATISTICA Text Miner:

- Contém várias opções de acesso a documentos textuais em diversos diferentes, incluindo TXT, PDF, PostScript, HTML, XML, RTF e DOC.
- Provê suporte à web-*crawling*, permitindo extrair documentos de páginas Web. Também é possível selecionar os documentos a partir de uma pasta local, conforme ilustrado na Figura 28.
- Inclui *stoplists* e algoritmos de *stemming* para as línguas: dinamarquês, holandês, inglês, francês, alemão, italiano, português, espanhol, sueco etc. As *stoplists* podem ser editadas manualmente pelo usuário. O software é projetado de modo que o suporte a línguas adicionais possa ser feito com o mínimo de esforço;
- Efetua a contagem de frequência de todas as palavras nos documentos, que serve de base para todas as análises numéricas subsequentes. Filtros adicionais podem ser aplicados. Por exemplo, cálculo de frequência normalizada dos termos, frequência inversa dos documentos, gerando uma sumarização numérica dos documentos;
- Disponibiliza métodos de análise estatística que são aplicadas sobre os sumários numéri¬cos dos documentos, técnicas de Clusterização, tais como *k-Means* para identificar documentos similares relacionamentos entre estes.

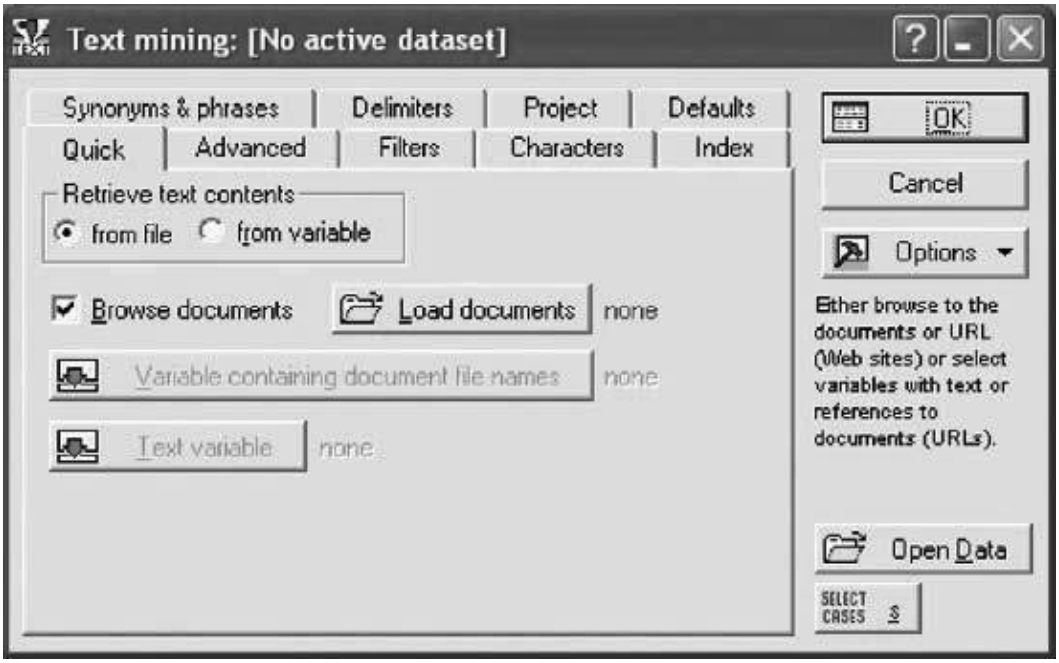

Figura 28 – Interface de coleta do software STATISCA