UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

PAULO HENRIQUE DA SILVA

Classificação Automática de Documentos: seleção customizada do classificador

Goiânia
2020

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[x] Dissertação [ ] Tese

2. Nome completo do autor

Paulo Henrique da Silva

3. Título do trabalho

Classificação Automática de documentos: seleção customizada do classificador

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Wellington Santos Martins**, **Professor do Magistério Superior**, em 21/12/2020, às 16:22, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **PAULO HENRIQUE DA SILVA**, **Discente**, em 21/12/2020, às 16:33, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1767952** e o código CRC **B3644899**.

Referência: Processo nº 23070.047793/2020-54 SEI nº 1767952

PAULO HENRIQUE DA SILVA

Classificação Automática de Documentos: seleção customizada do classificador

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Programa de Pós-Graduação em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Wellington Santos Martins

Goiânia
2020

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

da Silva, Paulo Henrique
Classificação Automática de Documentos: seleção customizada do classificador [manuscrito] / Paulo Henrique da Silva. - 2020.
LXXX, 80 f.: il.

Orientador: Prof. Dr. Wellington Santos Martins.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2020.
Bibliografia.
Inclui tabelas, lista de figuras, lista de tabelas.

1. Classificação Automática de Documentos. 2. Conjunto de Classificadores. 3. Seleção Dinâmica do Classificador. 4. Programação Paralela. I. Santos Martins, Wellington, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº **26/2020** da sessão de Defesa de Dissertação de **Paulo Henrique da Silva**, que confere o título de Mestre em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos vinte e três dias do mês de novembro de dois mil e vinte, a partir das catorze horas e trinta minutos, via sistema de webconferência da RNP, realizou-se a sessão pública de Defesa de Dissertação intitulada **"Classificação Automática de documentos: seleção customizada do classificador"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor Wellington Santos Martins (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professor Doutor Daniel Xavier de Sousa (IFG), membro titular externo; Professor Doutor Thierson Couto Rosa (INF/UFG), membro titular interno. A realização da banca ocorreu per meio de videoconferência, em atendimento à recomendação de suspensão das atividades presenciais na UFG emitida pelo Comitê UFG para o Gerenciamento da Crise COVID-19, bem como à recomendação de isolamento social da Organização Mundial de Saúde e do Ministério da Saúde para enfrentamento da emergência de saúde pública decorrente do novo coronavírus. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Wellington Santos Martins, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos vinte e três dias do mês de novembro de dois mil e vinte.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Thierson Couto Rosa**, **Professor do Magistério Superior**, em 23/11/2020, às 17:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Wellington Santos Martins**, **Professor do Magistério Superior**, em 23/11/2020, às 17:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Daniel Xavier de Sousa**, **Usuário Externo**, em 23/11/2020, às 17:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **PAULO HENRIQUE DA SILVA**, **Discente**, em 23/11/2020, às 17:34, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Referência: Processo nº 23070.047793/2020-54 SEI nº 1658151

PAULO HENRIQUE DA SILVA

Classificação Automática de Documentos: seleção customizada do classificador

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Programa de Pós-Graduação em Ciência da Computação, aprovada em 23 de Novembro de 2020, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Wellington Santos Martins
Instituto de Informática – UFG
Presidente da Banca

Prof. Dr. Thierson Couto Rosa
Instituto de Informática – UFG

Prof. Dr. Daniel Xavier de Sousa
Instituto Federal de Goiás – IFG

Paulo Henrique da Silva

Graduado em Ciências da Computação pela PUC–Goiás - Pontifícia Universidade Católica de Goiás, com especialização em Análise e Projeto de Sistemas da Informação pela UFG - Universidade Federal de Goiás. Atualmente, no mestrado, participa de grupo de pesquisa na linha de Sistemas de Computação com foco em similaridade de documentos e paralelismo.

Dedico este trabalho à minha família pelo apoio ao longo do período de pequisa e elaboração desta dissertação.

Agradecimentos

Agradecimentos ao Prof. Dr. Wellington Santos Martins orientador desta dissertação e a todos os participantes do grupo de pesquisa LDA. Também agradeço à instituição Universidade Federal de Goiás e a todos os seus coloboradores, em especial o corpo docente, que viabilizam a realização do programa de Pós-Graduação.

Não existe um caminho para a felicidade. A felicidade é o caminho.

Mahatma Gandhi,

.

Resumo

da Silva, Paulo Henrique. **Classificação Automática de Documentos: seleção customizada do classificador**. Goiânia, 2020. 80p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

O recente aumento nos dados armazenados digitalmente estimulou o desenvolvimento de métodos para organizar e extrair conhecimento relevante desse grande volume de dados. A classificação automática de documentos (ADC) é um desses métodos. Considerada uma das tarefas mais relevantes e desafiadoras no contexto de recuperação de informações, devido a alta dimensionalidade e esparsidade dos dados, utiliza técnicas de aprendizado de máquina para agrupar documentos similares em classes. Trabalhos recentes defendem o uso de sistemas de múltiplos classificadores (MCS) para aumentar a precisão da ADC, através da combinação de um conjunto de classificadores para obter melhores resultados em relação a um único classificador. Uma das abordagens mais promissoras de MCS é a seleção dinâmica (DS), onde os classificadores base são selecionados em tempo real, de acordo com cada novo documento de consulta (teste) a ser classificado. Este trabalho propõe a seleção customizada de método de classificação realizada em tempo de consulta (teste). Somente o classificador mais competente, ou o conjunto de classificadores mais competentes, é selecionado para fazer a predição do rótulo de cada documento de consulta. Alem disso, o trabalho apresenta a exploração de paralelismo para acelerar a tarefa de ADC. Resultados experimentais, utilizando bases de dados padronizadas, mostram resultados competitivos e promissores em relação às *baselines* usadas. Novas oportunidades para exploração de paralelismo também são apresentadas como trabalhos futuros.

Palavras–chave

Classificação Automática de Documentos, Conjunto de Classificadores, Seleção Dinâmica do Classificador, Programação Paralela

Abstract

da Silva, Paulo Henrique. . Goiânia, 2020. 80p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

The recent increase in digitally stored data has spurred the development of methods to organize and extract relevant knowledge from this large volume of data. Automatic document classification (ADC) is one such method. Considered one of the most relevant and challenging tasks in the context of information retrieval, due to the high dimensionality and sparse data, it uses machine learning techniques to group similar documents into classes. Recent works advocate the use of multiple classifier systems (MCS) to improve the accuracy of ADC, through the combination of a set of classifiers to obtain better results in relation to a single classifier. One of the most promising approaches to MCS is dynamic selection (DS), where the base classifiers are selected in real time, according to each new consultation document (test) to be classified. This work proposes the customized selection of the classification method performed in consultation time (test). Only the most competent classifier, or the most competent set of classifiers, is selected to predict the label of each consultation document. In addition, the paper presents the exploration of parallelism to speed up the ADC task. Experimental results, using standardized databases, show competitive and promising results in relation to the baselines used. New opportunities for exploring parallelism are also presented as future work.

Keywords

Automatic Document Classification, Ensemble of Classifiers, Dynamic Classifier Selection, Parallel Programming

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Nos últimos anos a quantidade de informação gerada e armazenada em meios digitais vem crescendo a cada dia. De acordo com Lyman [Lyman e Varian 2004] esse crescimento tem ocorrido em escala exponencial. O processamento dessa grande quantidade de dados de forma eficiente é crítico para a recuperação de informações (RI). Este recente aumento nos dados armazenados digitalmente estimulou o desenvolvimento de métodos para organizar e extrair conhecimento relevante desse grande volume de dados. A classificação automática de documentos (ADC) é um desses métodos. Apesar do desenvolvimento de novas tecnologias na representação de texto (como por exemplo *embeddings*, *meta-características* e algoritmos de classificação), a classificação automática de documentos ainda é considerada uma tarefa relevante e desafiadora [Mendes L. F. 2020]]. A alta dimensionalidade e esparsidade dos conjuntos de dados usados pela ADC (vetores longos e esparsos) tornam essa tarefa desafiadora. Técnicas de aprendizado de máquina são geralmente empregadas para construir modelos de aprendizagem de máquina, baseados nas palavras (termos ou características) dos documentos, que sejam capazes de classificar novos documentos, ainda não conhecidos pelos modelos.

Por exemplo, se considerarmos mensagens de e-mail como documentos, é comum querermos classificar novas mensagens como spam ou não spam. Isso é feito considerando as palavras (termos ou características) presentes na mensagem, geralmente associando um vetor (do tamanho do vocabulário usado) com pesos (ex. frequência do termo) a cada documento. A classificação automática de documentos pode ser adotada em várias outras aplicações reais como sistemas de recomendação, análise de sentimentos, categorização de tópicos, entre outras. A ADC é um problema genérico não limitado apenas à classificação de texto, podendo também pode ser estendido para outros domínios como música, imagens, vídeo e outras mídias.

A tarefa de ADC pode ser implementada usando um classificador ou um conjunto de classificadores (*ensemble*) para decidir a classe de um determinado documento. Para atingir um melhor desempenho geralmente recorre-se à técnica de conjunto de classificadores, na qual vários algoritmos são combinados de maneira inteligente para obter melhores resultados. Trabalhos recentes defendem o uso de sistemas de múltiplos classifi-

cadores (MCS) para aumentar a precisão da ADC, através da combinação de um conjunto de classificadores para obter melhores resultados em relação a um único classificador [Cruz, Sabourin e Cavalcanti 2018] e [Jr, Sabourin e Oliveira 2014]. Uma das abordagens mais promissoras de MCS é a seleção dinâmica (DS), onde os classificadores base [1] são selecionados em tempo real, de acordo com cada novo documento de consulta (teste) a ser classificado. De acorco com [Jr, Sabourin e Oliveira 2014] e [Cruz et al. 2015], a seleção dinâmica tem desempenho superior sobre as abordagens tradicionais de combinação como *majority voting* e *Boosting*. As técnicas de DS estimam o nível de competência de cada classificador, de um conjunto de classificadores (*ensemble*), e selecionam o classificador ou conjunto de classificadores mais competente para prever o rótulo de um documento de consulta. A estimativa é baseada em uma região local do espaço de características onde o documento de consulta está localizado.

O presente trabalho contribui com uma nova estratégia para selecionar, de maneira customizada, o melhor classificador ou subconjunto de classificadores a ser utilizado quando o documento de consulta é apresentado para ser classificado. Este trabalho propõe uma nova maneira de se fazer a seleção customizada do método de classificação, realizada em tempo de consulta (teste). Somente o classificador mais competente, ou o conjunto de classificadores mais competentes, é selecionado para fazer a predição do rótulo de cada documento de consulta. Nosso método, denominado SMART, implementa três formas de *ensemble* com o(s) classificador(es) mais competente(s), selecionado(s) de acordo com a região local (vizinhança) do documento de consulta. Além disso, como a ADC requer um alto custo computacional, devido ao tamanho dos conjuntos de dados utilizados, este trabalho discute abordagens paralelas para acelerar o tempo de execução desta tarefa.

Para avaliar a efetividade do método proposto foram realizados experimentos, utilizando conjuntos de dados padronizadas, de duas aplicações de ADC, categorização de tópicos e análise de sentimentos. Comparamos nossa proposta com os métodos MetaLazy [Mendes L. F. 2020], Support Vector Machine (SVM) e Bidirectional Neural Network (BERT) [Devlin et al. 2018]. Os experimentos demonstram resultados competitivos e promissores em relação às *baselines* utilizadas.

As principais contribuições deste trabalho são:

- A proposta do método SMART, um *ensemble* para ADC com as seguintes novidades:
    1. Nosso método possibilita três formas de *ensemble*: votação majoritária, ponderação por acerto e ponderação por similaridade.

[1] O termo classificador básico se refere a um único classificador pertencente a um conjunto de classificadores.

2. Treinamento dos classificadores base, utilizando os conjuntos de dados de treino e validação, antes da etapa de consulta. Em outras abordagens da literatura, o treinamento é realizado utilizando os k vizinhos do documento de consulta como conjunto de dados de treino e na etapa de consulta;

- Uma extensiva experimentação com conjuntos de dados padronizados e a comparação com três *baselines*.

Esta dissertação é organizada da seguinte maneira. No capítulo 2 são descritos conceitos e algoritmos que servem de embasamento para este trabalho. No capítulo 3 é descrito o método proposto de seleção customizada do classificador (SMART), com experimentos e avaliações usados para comparar sua efetividade. No capítulo 4 são apresentadas conclusões finais, resultados alcançados e trabalhos futuros, assinalando possíveis direções para investigações futuras.

CAPÍTULO 2

Embasamento

2.1 Representação de documentos

2.1.1 Preparação dos dados

Manning (2010) [Manning, Raghavan e Schütze 2010] afirma que a recuperação de informação pode ser definida como a busca de documentos não estruturados, que satisfaçam uma necessidade de informação a partir de grandes coleções de documentos que geralmente são armazenadas em computadores.

Recuperação de informação (RI), de acordo com [Baeza-Yates e Ribeiro-Neto 2013], trata da representação, armazenamento, organização e acesso a itens de informação, como documentos, páginas Web, catálogos *online*, registros estruturados e semiestruturados, objetos multimídia, etc. Desta forma, ela deve permitir que usuários obtenham acesso à informação esperada de maneira facilitada.

Os sistemas de recuperação de informação lidam com a busca de documentos de acordo com a solicitação do usuário e seu objetivo principal é recuperar todos os documentos que são relevantes à necessidade do usuário. O processo de recuperação de informação é bastante complexo e de acordo com [Fayyad, Piatetsky-Shapiro e Smyth 1996] dividi-se nas seguintes etapas:

- Seleção dos dados;
- Pré-processamento dos dados;
- Transformação dos dados;
- Mineração dos dados;
- Avaliação dos dados;
- Apresentação dos dados.

Uma das principais dificuldades da recuperação de informações é que os documentos, dos conjuntos de dados utilizados, normalmente não estão em formato estruturado. Isto implica em uma limitação ao uso de algoritmos de aprendizado de máquina, já que esses algoritmos precisam de dados estruturados para que seja possível extrair co-

nhecimento relevante. A etapa de preparação (pré-processamento) dos dados é uma das principais etapas da recuperação de informações e é usada para transformar dados não estruturados em dados estruturados.

A etapa de pré-processamento consiste em um conjunto de operações, realizadas sobre os documentos, antes que estes sejam enviados ao algoritmo de aprendizagem de máquina. Um exemplo desta etapa pode ser simplificadamente dividida em quatro operações, conforme apresentada a seguir:

1. **Tokenização** - extração de todas as palavras dos documentos que compõem o conjunto de documentos (*corpus*). Consiste na partição do documento em palavras ou símbolos identificados como um termo (token).
2. **Remoção de Stopwords** - eliminação de palavras ou termos que não trazem relevância ou significado ao texto, quando analisadas separadamente, como exemplo podemos citar artigos, preposições e pronomes.
3. **Stemming** - consiste em reduzir as palavras ou termos a uma forma comum denominada *stem*, através da remoção de seus afixos (prefixos e sufixos).
4. **Taxonomia ou Tesaurus** - construção de estruturas de categorização de termos para filtrar termos relevantes ao processo e/ou para expandir determinado contexto de um documento adicionando termos relacionados e sinônimos.

A lista de operações aprentada acima não é exaustiva, sendo possível realizar outras operações para conseguir colocar os dados em um formato estruturado.

Existem várias maneiras de transformar documentos em dados estruturados, sendo que uma forma é transformá-los em uma representação atributo-valor utilizando a abordagem *bag of words*. Com essa forma de modificação dos dados é gerada uma tabela com a contagem da frequência de cada palavra (termo) em cada documento, independente do seu contexto ou significado. A tabela 2.1 apresenta uma representação da frequência de termos em documentos utilizando essa abordagem. Nessa tabela, cada documento d_i é representado por uma linha e as colunas representam as palavras (termos) do vocabulário dos documentos. Sendo que o vocabulário é composto por todas as palavras que existem nos documentos do conjunto de dados. O valor de cada atributo a_{NM} da tabela indica a frequência que a palavra (termo) aparace no documento, caso a palavra não ocorra no documento o valor do atributo será 0 (zero).

Tabela 2.1: Representação de documentos no formato atributo-valor

	t_1	t_2	t_3	...	t_M
d_1	a_{11}	a_{12}	a_{13}	...	a_{1M}
d_2	a_{21}	a_{22}	a_{23}	...	a_{2M}
d_3	a_{31}	a_{32}	a_{33}	...	a_{3M}
...	...	...	...	...	...
d_N	a_{N1}	a_{N2}	a_{N3}	...	a_{NM}

Uma tabela atributo-valor, gerada utilizando a abordagem *bag of words*, tem algumas particularidades que precisam ser observadas. Primeiro a alta dimensionalidade, devido ao grande número de palavras (vocabulário) utilizadas no conjunto de documentos. Segundo a esparsidade dos dados, o texto de cada documento pode conter apenas um subconjunto pequeno de todas as palavras existentes no conjunto de documentos (vocabulário).

Existem várias maneiras de associar valores a cada um dos termos em um documento, sendo as mais comuns a presença ou ausência da palavra, número absoluto de aparições das palavras (freqüência absoluta), ou a freqüência relativa ou ponderada dessas palavras em relação ao número de documentos.

2.1.2 TF-IDF (*Term Frequency - Inverse Document Frequency*)

Uma maneira de representar os documentos, de um determinado conjunto de documentos, é através da ponderação de termos presentes nesse conjunto. A frequência do termo (TF - *Term Frequency*) e a frequência inversa de documento (IDF - *Inverse Docment Frequency*) são os fundamentos da medida de ponderação mais utilizada, em recuperação de informações, para representação de documentos. [Baeza-Yates e Ribeiro-Neto 2013]. O seu cálculo está baseado em dois pressupostos:

- Quanto maior é a frequência de um termo em um documento, maior é a sua relevância no documento;
- Quanto maior é a ocorrência de um termo nos documentos de uma coleção, menor é a sua especificidade ou discriminante entre os documentos.

Enquanto o *tf* representa a frequência de um determinado termo no documento normalizada pela quantidade total de termos do documento, o *idf* representa a especificidade do termo em relação ao conjunto de documentos (inverso do número de documentos nos quais o termo ocorre). Abaixo é apresentada uma equação para o cálculo do peso *tf-idf*.

$$w_{i,j} = \frac{n_{i,j}}{\sum_{i=1}^{k} n_{k,j}} \; x \; \log \frac{|D|}{|d_j : t_i \in d_j|} \tag{2-1}$$

Onde $n_{i,j}$ é o número de ocorrências do termo i no documento d_j e o denominador é o número de ocorrências de todos os termos do documento d_j. E $|D|$ é o número de documentos no conjunto de documentos (*corpus*) e $|d_j : t_i \in d_j|$ é o número de documentos que contém o termo t_i.

A fórmula do *tf-idf* pode ser estendida para comparar a similaridade entre dois documentos d_i e d_j ou entre um documento d_i e um documento de consulta q. Isto é feito simplesmente expressando todos os valores de *tf-idf* dos termos em comum entre os dois objetos como um produto escalar entre os vetores formados por esses dois objetos, conforme pode ser visto na equação 2-2.

Para exemplificar o cálculo do *tf-idf* primeiro consideramos o exemplo de um conjunto de documentos retirado do livro Recuperação de Informação dos autores [Baeza-Yates e Ribeiro-Neto 2013].

Tabela 2.2: Coleção de documentos

d_1	To do is to be. To be is to do.
d_2	To be or not to be. I am what I am.
d_3	I think therefore I am. Do be do be do.
d_4	Do do do, da da da. Let it be, let it be.

Com o conjunto de documentos definido na tabela 2.2 podemos fazer o cálculo do *tf-idf* de cada termo em cada um dos documentos do conjunto de documentos. Na tabela 2.3 é apresentado o resultado do cálculo do *tf-idf*. Onde n_i é o número de documentos que contém o termo i, *tf* é a frequência do termo i no documento e *idf* é o inverso da frequência do termo i no conjunto de documentos.

Tabela 2.3: Cálculo do tf-idf.

termo	n_i	$tf_{i,1}$	$tf_{i,2}$	$tf_{i,3}$	$tf_{i,4}$	idf_i	$tfidf_{i,1}$	$tfidf_{i,2}$	$tfidf_{i,3}$	$tfidf_{i,4}$
to	2	3	2	-	-	1	3	2	-	-
do	3	2	-	2,585	2,585	0,415	0,830	-	1,073	1,073
is	1	2	-	-	-	2	4	-	-	-
be	4	2	2	2	2	0	-	-	-	-
or	1	-	1	-	-	2	-	2	-	-
not	1	-	1	-	-	2	-	2	-	-
I	2	-	2	2	-	1	-	2	1	-
am	2	-	2	1	-	1	-	2	-	-
what	1	-	1	-	-	2	-	2	-	-
think	1	-	-	1	-	2	-	-	2	-
therefore	1	-	-	1	-	2	-	-	2	-
da	1	3	-	-	2,585	2	-	-	-	5,170
let	1	-	-	1	2	2	-	-	-	4
it	1	-	-	1	2	2	-	-	-	4

2.1.3 Modelo vetorial

Para realizar a tarefa de classificação automática de documentos é preciso definir um modelo de representação dos documentos do conjunto de dado utilizados. O modelo de representação é fundamental para que o algoritmo utilizado na ADC seja capaz de extrair conhecimento relevante do conjunto de dados. Entre esses modelos destacam-se o modelo booleano, o modelo vetorial e o modelo probabilístico. Neste trabalho será apresentado apenas o modelo vetorial.

O modelo vetorial é um modelo estatístico, em que um documento é representado como um vetor de termos em um espaço euclidiano *n-dimensional*, onde *n* é o número de termos, no qual cada dimensão representa um termo extraído do documento em questão. Cada termo pode ter um peso associado, e este peso ajuda a discretizar os documentos do conjunto de dados. Essa atribuição de pesos deve refletir a importância de cada termo no contexto do documento e também na coleção de documentos.

A posição do documento em cada dimensão é dada pelo peso do termo associado àquela dimensão. Portanto, a distância entre dois documentos indica o grau de similaridade entre ambos, ou seja, documentos com os mesmos termos estão localizados em uma mesma região do espaço e, teoricamente, possuem conteúdos similares.

A atribuição de pesos não binários aos termos do documento de consulta e aos termos dos documentos do conjunto de documentos (*corpus*) possibilita o cálculo do grau

de similaridade entre o documento de consulta e os documentos do conjunto de documentos. Usando uma ordenação decrescente dos documentos, levando em consideração o seu grau de similaridade, o modelo vetorial é capaz de medir a similaridade entre o documento de consulta e os documentos do *corpus*, fornecendo uma resposta mais precisa para a consulta realizada.

Para o modelo vetorial, o peso $w_{i,j}$ associado ao par termo-documento (t_i, d_j) é um valor não negativo e não binário e $w_{i,q}$ é o peso associado ao par termo-consulta (t_i, q) com $w_{i,q} \geq 0$. Os termos são representados por vetores unitários em um espaço com n dimensões, no qual n é o número de termos de indexação. As representações do documento d_j e do documento de consulta q são vetores com n dimensões dadas por:

$$\tilde{d_j} = (w_{1,j}, w_{2,j}, ..., w_{n,j})$$
$$\tilde{q} = (w_{1,q}, w_{2,q}, ..., w_{n,q})$$

Um documento d_j e um documento de consulta q são representados como vetores de n dimensões. O modelo vetorial calcula o grau de similaridade entre o documento d_j e o documento de consulta q com a correlação entre os vetores $\tilde{d_j}$ e $\tilde{q}$. Uma maneira de quantificar essa correlação é pelo cosseno do ângulo entre esses dois vetores, dada pela equação a seguir.

$$sim(d_j, q) = \frac{\tilde{d_j} \; x \; \tilde{q}}{\|\tilde{d_j}\| \; x \; \|\tilde{q}\|} \tag{2-2}$$

$$sim(d_j, q) = \frac{\sum_{i=1}^{n} w_{i,j} \; x \; w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} x \sqrt{\sum_{i=1}^{n} w_{i,q}^2}} \tag{2-3}$$

Similaridade do cosseno

Onde $\|\tilde{d_j}\| \; x \; \|\tilde{q}\|$ são as normas dos vetores e $\tilde{d_j} \; x \; \tilde{q}$ é o produto escalar entre os vetores.

O fator $||\tilde{d_j}||$ faz a normalização pelo tamanho do documento. Como $w_{i,j} \geq 0$, $w_{i,q} \geq 0$ e $sim(d_j, q)$ varia entre 0 e 1, isso faz com que o modelo vetorial ordene os documento de acordo com o grau de similaridade em relação ao documento de consulta. Os pesos mais utilizados no modelo vetorial são os calculados com a medida *tf-idf*, sendo aplicados para valores de frequência de termo maior do que zero.

2.2 Classificação de documentos

Com o aumento na quantidade de dados disponíveis em meios digitais, organizar e extrair informações úteis desses dados tornou-se uma tarefa importante para a indústria e a sociedade [Campos et al. 2017]. Ao usar técnicas de aprendizado de máquina para associar automaticamente documentos a classes, a classificação automática de documentos (ADC) fornece meios para organizar informações, permitindo melhor compreensão e interpretação dos dados [Baeza-Yates e Ribeiro-Neto 2013]. Muitas aplicações do mundo real, como categorização de tópicos, análise de sentimento, filtragem de *spam*, identificação de idioma, sistemas de recomendação, entre outras, podem ser resolvidas de forma eficaz e eficiente utilizando a classificação automática de documentos.

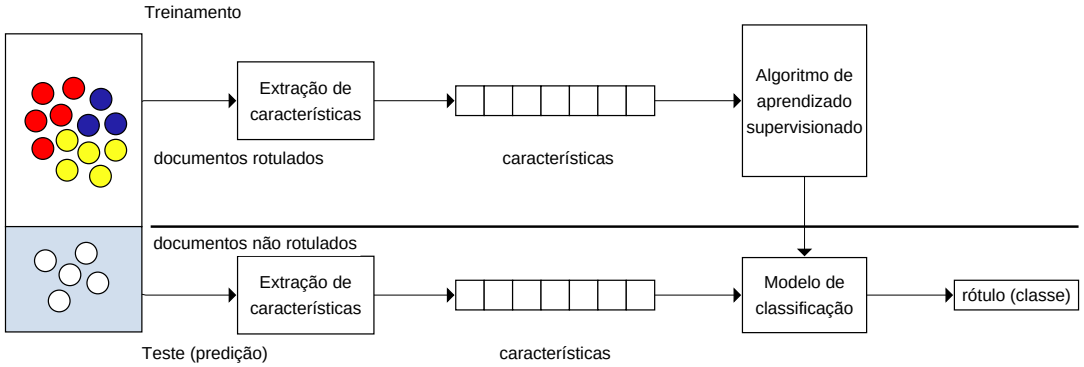
Considerado um dos problemas mais relevantes e desafiadores, a classificação ou categorização de documentos, envolve organizar documentos em classes ou categorias, com base nas características ou termos de cada documento. Para [Sarkar 2016], o problema de ADC torna-se mais desafiador à medida que o número de documentos, do conjunto de dados, a serem classificados, aumenta para centenas de milhares ou milhões. Neste momento, a aplicação de técnicas como extração de características e aprendizado de máquina são utilizadas para automatizar esta tarefa.

De acordo com [Canuto et al. 2018], a ADC pode ser definida da seguinte maneira: dado um conjunto de documentos de treinamento, a tarefa da classificação automática de documentos é aprender a classificar novos documentos automaticamente (não classificados), usando uma combinação de recursos desses documentos que os associa a uma ou mais categorias (classes).

O aprendizado de máquina pode ser dividido em aprendizado supervisionado, aprendizado não supervisionado, aprendizado por reforço. Cada um desses conceitos envolve várias técnicas e algoritmos que podem ser aplicados nos conjuntos de dados (documentos) para construir sistemas (modelos) de aprendizagem. Um modelo de aprendizado de máquina é uma combinação de dados e algoritmos. Uma das vantagens do aprendizado de máquina é que, após a construção (treinamento) do modelo, pode-se utilizá-lo diretamente em novos dados, que nunca foram vistos pelo classificador, para extrair características relevantes destes dados e alcançar uma boa efetividade do modelo. No presente trabalho será tratado apenas o aprendizado supervisionado.

Um classificador é definido como supervionado quando é construído baseado no conjunto de dados de treinamento, que contém os rótulos (classes) corretos para cada documento do conjunto de dados. O modelo aprendido é aplicado para fazer a predição de novos documentos não rotulados, ainda não conhecidos pelo classificador, conforme apresentado na figura 2.1.

Figura 2.1: Aprendizado Supervisionado



2.2.1 Classificação

2.2.1.1 Grade de pesquisa (GridSearch)

Para construir um modelo de classificação para um determinado algoritmo de aprendizado de máquina é necessário definir vários parâmetros para que o classificador obtenha o melhor desempenho possível. Estes parâmetros não são aprendidos diretamente pelos classificadores e são chamados de hiperparâmetros. No aprenzidado de máquina, um hiperparâmetro é um parâmetro cujo valor é utilizado para controlar o aprendizado do algoritmo. Como exemplo, podemos citar a taxa de aprendizado em uma rede neural, porque o valor deste hiperparâmetro é informado antes dos dados de treinamento serem alimentados no modelo.

De acordo com [Bergstra e Bengio 2012] o problema de identificar bons valores para hiperparâmetros é chamado de problema de otimização de hiperparâmetros, e a etapa crítica na otimização destes hiperparâmetros é a escolha do conjunto de ensaios (conjunto de valores que serão usados em cada parâmetro). Ainda segundo [Bergstra e Bengio 2012], a estratégia mais amplamente utilizada é uma combinação de *GridSearch* e pesquisa manual [LeCun et al. 1998], [Larochelle et al. 2007] e [Nair e Hinton 2010].

O *GridSearch* é uma técnica que realiza um ajuste dos valores dos hiperparâmetros para identificar os valores ideais para um determinado modelo de aprendizado de máquina. É uma pesquisa exaustiva realizada sobre os valores de hiperparâmetros especificados para um modelo de aprendizado, onde todas as combinações possíveis de valores dos hiperparâmetros são avaliadas e a melhor combinação é selecionada. Como o desempenho de todo o modelo é baseado nos hiperparâmetros especificados, então a descoberta do modelo de parâmetros ideal é geralmente uma tarefa crucial em aplicações de classificação [Jiménez, Lázaro e Dorronsoro 2009].

Segundo [Bergstra e Bengio 2012], se Λ for um conjunto indexado por K variáveis de configuração (por exemplo, para o algoritmo *SVM* seria o parâmetro de regularização C, ou o número máximo de iterações max_iter, que o algoritmo realiza para obter seus resultados), então o *GridSearch* requer a escolha de um conjunto de valores para cada variável ($L^{(1)}...L^{(k)}$). No *GridSearch*, o conjunto de tentativas é formado pela montagem de todas as combinações possíveis de valores, então o número de tentativas em um *GridSearch* é $S = \prod_{k=1}^{K} |L^{(k)}$ elementos. Este produto sobre K conjuntos faz com que o *GridSearch* sofra com a alta dimensionalidade porque o número de combinações de valores cresce exponencialmente com o número de hiperparâmetros [Bellman 1961]. A busca manual é usada para identificar regiões em Λ que são promissoras e para desenvolver a intuição necessária para escolher os conjuntos $L^{(k)}$ [Bergstra e Bengio 2012].

Os parâmetros selecionados são aqueles que maximizam a pontuação dos dados deixados de fora, a menos que uma pontuação explícita seja passada, caso em que ela é usada. O desempenho dos hiperparâmetros selecionados e do modelo treinado é então medido em um conjunto de dados de avaliação (teste ou validação) que não foi usado durante a etapa de seleção do modelo.

2.2.1.2 Validação cruzada

De acordo com [Krstajic et al. 2014] em uma situação ideal, teríamos dados suficientes para treinar (amostras de treinamento) e validar nossos modelos (amostras de validação) e ter dados separados para avaliar a qualidade de nossos modelos (amostras de teste). Tanto as amostras de treinamento quanto as amostras de teste precisam ser suficientemente grandes e diversificadas para serem representativas. No entanto, essas situações ricas em dados são raras nas pesquisas. Um grande problema com a seleção e avaliação de modelos é que geralmente temos apenas informações das amostras de treinamento e, portanto, não é viável calcular um erro de teste. No entanto, embora não possamos calcular o erro de teste, é possível estimar o erro de teste esperado usando amostras de treinamento.

[Stone 1974] defende a aplicação de um critério de validação cruzada para a escolha e avaliação da previsão estatística. Consiste na divisão controlada ou não controlada da amostra de dados em duas subamostras (subamostras A e B), a escolha de um preditor estatístico, incluindo qualquer estimativa necessária, na subamostra A. Em seguida, a avaliação de seu desempenho medindo suas predições na subamostra B.

Neste caso, o conjunto de treinamento é dividido em k partes, cada uma composta por l/k amostras. $k-1$ partes são usadas como um conjunto de treinamento e a parte restante é usada como um conjunto de validação ou teste. Um modelo estatístico é reajustado K vezes com os dados de cada partição do conjunto de treinamento. A análise do

desempenho do algoritmo é realizada na parte que foi separada para teste ou validação. A avaliação final é obtida através da média entre as k avaliações realizadas.

Para [Campos et al. 2017], o conjunto de treinamento original D_{train} é dividido em K conjuntos disjuntos de tamanhos iguais. Cada conjunto é composto por l/k amostras. $k-1$ conjuntos são usados como dados de treinamento e o restante é usado como dados de validação ou teste. Cada classificador base aprende considerando $D_{train} \setminus F_k$ como conjunto de treinamento e reservando F_k para teste. Formalmente, $\forall i = 1, ..., T: \forall k = 1, ..., K : C_i^k = L_i(D_{train} \setminus F_k)$. Subseqüentemente, cada classificador aprendido C_i^k produz uma estimativa para a probabilidade posterior. Dessa forma, segundo [Krstajic et al. 2014], um modelo estatístico é reajustado K vezes com os dados de cada partição do conjunto de treinamento. A análise do desempenho do algoritmo é realizada na partição que foi separada para teste ou validação. A avaliação final é obtida através de uma média das k avaliações realizadas.

Como [Stone 1974] definiu, a validação cruzada pode ser usada para seleção e avaliação de modelo, mas as duas tarefas requerem abordagens de validação cruzada diferentes. Segundo [Krstajic et al. 2014] a validação cruzada pode ser dividida em dois tipos:

- Na validação cruzada de *k folds* (partes ou dobras), o conjunto de dados é dividido aleatoriamente em k dobras, e um modelo estatístico é reajustado k vezes, com os casos de cada dobra mantidos sucessivamente a partir do conjunto de treinamento. É analisada a variação no desempenho de predição que resulta da escolha de uma divisão diferente dos dados;
- Validação cruzada estratificada de *k folds* (partes ou dobras), a variável de saída é primeiro estratificada e o conjunto de dados é aleatoriamente dividido em k dobras, garantindo que cada dobra contenha aproximadamente a mesma proporção para cada classe presente no conjunto de documentos.

A validação cruzada pode ser aplicada para as fases de treinamento e teste em problemas de classificação e para ajuste de parâmetros dos classificadores (*GridSearch*). [Chang e Lin 2011] sugerem a escolha de um conjunto inicial de parâmetros de entrada e a realização de validação cruzada em grade de pesquisa (*gridsearch cross-validation*), para encontrar os parâmetros ideais (com relação à grade fornecida e ao critério de pesquisa fornecido), para os algoritmos em análise. A validação cruzada em grade de pesquisa produz estimativas de desempenho estatístico (por exemplo, taxa de erro) para cada valor na grade.

2.2.1.3 Treino

O problema de classificação tem sido amplamente estudado nas comunidades de mineração de dados e recuperação de informações [Aggarwal e Zhai 2012]. [Sebastiani 2002] definiu que a abordagem de aprendizado de máquina depende da disponibilidade de um conjunto de dados inicial $D = \{d_1, \ldots, d_{|D|}\}$ e de documentos pré-classificados em $C = \{c_1, \ldots, c_{|C|}\}$. Ou seja, os valores da função $\Phi : DxC \rightarrow \{T, F\}$ são conhecidos para cada par $\langle d_j, c_i \rangle \in DxC$. Um documento d_j é um exemplo positivo de c_i se $\Phi(d_j, c_i) = T$, e um exemplo negativo de c_i se $\Phi(d_j, c_i) = F$.

Em um nível mais detalhado, um classificador é uma função que encontra os valores de várias características do documento e prevê a classe à qual este documento pertence [Pereira, Mitchell e Botvinick 2009]. Treinamento é o processo em que o algoritmo de aprendizado supervisionado analisa e tenta inferir padrões a partir do conjunto de dados de treinamento, de forma que possa identificar quais padrões levam a um resultado específico, [Sarkar 2016]. Esses resultados são conhecidos como rótulos de classes. Normalmente é realizado o processo de extração ou engenharia de características (pré-processamento) para obter características relevantes dos dados brutos antes do treinamento. Esses conjuntos de características são fornecidos a um algoritmo específico, que busca identificar e aprender padrões com esses dados e seus resultados correspondentes (rótulos de classes). O resultado é uma função inferida conhecida como modelo de classificação. Espera-se que este modelo seja generalizado o suficiente, a partir dos padrões de aprendizagem no conjunto de treinamento, de modo que possa prever as classes ou resultados para novos dados não conhecidos. Os dados de treinamento são usados para construir um modelo de classificação, que relaciona as características do documento a um dos rótulos de classe [Aggarwal e Zhai 2012].

Segundo [Aggarwal e Zhai 2012], quando um modelo de aprendizado de máquina é treinado, o objetivo não é apenas que o algoritmo aprenda a modelar os dados de treinamento. É necessário que o algoritmo seja capaz de generalizar para dados que nunca viu antes. Existe uma maneira apropriada de medir o desempenho da generalização de um algoritmo, essa maneira é medir seu desempenho em um conjunto de dados de teste ou validação, que é composto por dados que o algoritmo ainda não conhece. Se um algoritmo funciona bem no conjunto de treinamento, mas não consegue generalizar, dizemos que há um ajuste excessivo (*overfitting*). Melhorar a generalização (ou prevenir *overfitting*) em algoritmos de aprendizado de máquina é uma tarefa difícil, mas existem técnicas que auxiliam na melhoria da generalização, como por exemplo:

- Separar o conjunto de dados em conjunto de treino, conjunto de validação e conjunto de teste e conhecer as diferenças entre esses conjuntos;

- Entender como o treinamento e a função de erro funcionam no modelo de aprendizagem;
- Medir a generalização, otimizar a função de perda (ou função de custo) no treinamento do modelo de classificação.

2.2.1.4 Teste

No aprendizado de máquina, é comum coletar amostras disjuntas (independentes) de um conjunto de dados base para construir e ajustar modelos preditivos (supervisionados) [Mitchell et al. 1997]. A abordagem mais comum é chamada de "treinar e testar". A ideia básica é construir um modelo preditivo com uma amostra de treinamento e, em seguida, validar o modelo usando uma amostra de teste independente. Este processo tem como objetivo fornecer uma estimativa realista de precisão do modelo e alcançar a generalização quando o modelo é usado em novos dados. Essa abordagem é aplicada dividindo o conjunto de dados D em dois subconjuntos separados. O primeiro subconjunto é chamado de treino e o segundo subconjunto de teste.

Em ambientes de pesquisa (e na maioria dos ambientes operacionais também), uma vez que um classificador Φ tenha sido construído, é desejável avaliar sua eficácia. Neste caso, antes da construção do classificador, o conjunto de dados inicial é dividido em dois subconjuntos, não necessariamente de tamanhos iguais [Sebastiani 2002]:

- um conjunto de treinamento (e validação) $TV = \{d1, \ldots, d_{|TV|}$. O classificador Φ para as classes $C = \{c_1, \ldots, c_{|C|}\}$ é indutivamente construído observando as características destes documentos;
- um conjunto de teste $Te = \{d_{|TV|+1}, \ldots, d_{|D|}$, usado para testar a eficácia dos classificadores. Cada documento $d_j \in Te$ é alimentado para o classificador, e as decisões do classificador $\Phi(d_j, c_i)$ são comparadas com os rótulos reais de cada documento do conjunto. Uma medida da eficácia da classificação é baseada na frequência com que os predições de $\Phi(d_j, c_i)$ correspondem aos rótulos reais dos documentos do conjunto Te.

Os documentos em Te não podem participar na construção dos classificadores. Se esta condição não for satisfeita, os resultados experimentais obtidos provavelmente seriam irrealisticamente bons, e a avaliação não teria, portanto, caráter científico [Mitchell et al. 1997].

Para [Sarkar 2016] a etapa de teste envolve a avaliação do desempenho das predições do modelo construído (aprendido) para verificar quão bem ele foi treinado e quanto foi aprendido, em termos de acurácia, no conjunto de dados de treinamento. Para isso, geralmente é usado um conjunto de dados de validação ou de teste. O desempenho do modelo é testado fazendo predições nesse conjunto de dados, e comparando as predições

realizadas com os rótulos (classes) reais dos documentos deste conjunto. Freqüentemente, também é usada a validação cruzada, em que os dados são divididos em partições (*folds*). Uma parte dos dados é usada para treinamento, e o restante usado para validar o modelo treinado. Podem ser feitos ajustes no modelo, com base nos resultados da validação, para obter uma configuração ideal que produza precisão máxima e erro mínimo.

O conjunto de dados de teste precisa ser uma amostra representativa de como as novas amostras de dados reais podem parecer, para as quais o modelo irá gerar predições e como ele pode se desempenhar nesses novas amostras de dados. Com o modelo de classificação construído podem ser utilizadas várias métricas para avaliar a sua performance. Algumas métricas de avalição são descritas na subseção 2.2.1.5.

2.2.1.5 Métricas de avaliação

Após realizar o treinamento, o ajuste e a construção de modelos de aprendizagem é preciso medir o seu desempenho. O desempenho dos modelos de aprendizado de máquina geralmente é baseado em quão bem eles prevêem os resultados para novos documentos. Normalmente, esse desempenho é medido em relação a um conjunto de dados de teste ou conjunto de dados de validação, que consiste em dados que não foram usados para treinar o classificador, ou seja, dados não conhecidos pelo modelo de aprendizagem. Este conjunto de dados de teste ou validação tem os rótulos correspondentes de cada documento para que possa ser comparado o rótulo (predição), atribuído pelo classificador, com o rótulo original do documento. Isto é feito para mensurar quanto o classificador acertou da predição destes documentos.

De acordo com [Hossin e Sulaiman 2015], para problemas de classificação binária, a avaliação da classificação pode ser definida com base na matriz de confusão, conforme mostrado na Tabela 2.4. A linha da tabela representa a classe prevista para o documento de consulta, enquanto a coluna representa a classe real deste documento. Considere, *tp* e *tn* o número de instâncias de documentos positivos e negativos classificadas corretamente; *fp* e *fn* o número de instâncias de documentos positivas e negativas classificadas incorretamente. A partir da Tabela 2.4, várias métricas podem ser calculadas, conforme apresentado na tabela 2.5, para avaliar o desempenho do classificador. Essas métricas podem ser adaptadas para avaliar problemas de classificação multi-classes.

Tabela 2.4: Matriz de confusão para classificação binária, pode ser adaptada para problemas de multi-classes (tabela adaptada de [Hossin e Sulaiman 2015]).

	Classe atual positiva	**Classe atual negativa**
Predição classe positiva	*tp*	*fn*
Predição classe negativa	*fp*	*tn*

Tabela 2.5: Métricas para avaliação de tarefas de classificação (tabela adaptada de [Hossin e Sulaiman 2015]).

Métrica	Fórmula	Descrição
Acurácia (a)	$\frac{tp+tn}{tp+fp+tn+fn}$	Mede a proporção de predições corretas sobre o número total de instâncias avaliadas.
Erro (e)	$\frac{fp+fn}{tp+fp+tn+fn}$	Mede a proporção de predições incorretas sobre o número total de instâncias avaliadas.
Precisão (p)	$\frac{tp}{tp+fp}$	Mede a proporção de documentos pertencentes à classe positiva sobre todas as predições de classes positivas feitas pelo classificador.
Revocação (r)	$\frac{tp}{tp+tn}$	Mede a proporção de documentos positivos que são classificados corretamente
Métrica F e F_1	$2\frac{p(c_i).r(c_i)}{p(c_i)+r(c_i)}$	Média harmônica entre precisão e revocação para uma determinada classe.
microF1	$2\frac{p.r}{p+r}$	Média aritmética simples da métrica F1 entre todas as classes do conjunto de dados.
macroF1	$\frac{1}{c}\sum_{i=1}^{c} 2\frac{p(c_i).r(c_i)}{p(c_i)+r(c_i)}$	Mede a proporção dos documentos classificados corretamente em cada classe.

2.2.2 Classificadores - SVM, SGD, Logistic Regression, RF, NB e kNN

2.2.2.1 SGD

Em um nível teórico, o gradiente descendente é um algoritmo que minimiza funções. Dada uma função definida por um conjunto de parâmetros, o gradiente descendente começa com um conjunto inicial de valores de parâmetros e se move iterativamente em direção a um conjunto de valores de parâmetros que encontram o ponto mínimo para a função. Essa minimização iterativa é obtida por meio de cálculo, dando passos na direção negativa do gradiente da função para encontrar seu mínimo [Prasetijo et al. 2017].

Cada exemplo z é um par (x, y) composto de uma entrada arbitrária x e uma saída escalar y. Consideramos uma função de perda $l(\hat{y}, y)$ que mede o custo de prever $\hat{y}$ quando a resposta real é y, e escolhemos um conjunto F de funções $f_w(x)$ parametrizadas por um vetor de peso w. Procuramos a função $f \in F$ que minimize a perda $Q(z, w) = l(f_w(x), y)$, calculada pela média dos exemplos $z_1 ... z_n$ [Bottou 2010].

$$E(f) = \int l(f(x), y) dP(z) \tag{2-4}$$

$$E_n(f) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i), y_i) \tag{2-5}$$

O risco empírico $E_n(f)$ mede o desempenho do conjunto de treinamento. O risco esperado $E(f)$ mede o desempenho de generalização, ou seja, o desempenho esperado

em exemplos futuros. A teoria da aprendizagem estatística [Chervonenkis e Vapnik 1971] justifica minimizar o risco empírico em vez do risco esperado quando o conjunto de funções F escolhido é suficientemente restritivo.

O gradiente descendente (GD) tem sido frequentemente proposto para minimizar o risco empírico $E_n(f_w)$. Cada iteração atualiza os pesos w com base no gradiente de $E_n(f_w)$

$$W_{t+1} = W_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_w Q(z_i, W_t) \tag{2-6}$$

onde γ é a taxa de aprendizagem escolhida adequadamente. Sob suposições de regularidade suficiente, quando a estimativa inicial w_0 está próxima o suficiente do ótimo, e quando a taxa de aprendizagem γ é suficientemente pequena, este algoritmo atinge convergência linear [Bottou 2010], ou seja, $-log\ p \sim t$, onde p representa o erro residual.

O algoritmo de gradiente descendente estocástico (SGD) é uma simplificação do gradiente descendente. Em vez de calcular o gradiente de $E_n(f_w)$ para todas as instâncias de z, cada iteração estima esse gradiente com base em um único exemplo escolhido aleatoriamente z_t:

$$W_{t+1} = W_t - \gamma_t \nabla_w Q(z_i, W_t) \tag{2-7}$$

O processo estocástico $\{w_t, t = 1, \ldots\}$ depende dos exemplos escolhidos aleatoriamente em cada iteração. Espera-se que 2-7 se comporte como sua expectativa 2-6 apesar do ruído introduzido por este procedimento simplificado [Bottou 2010].

2.2.2.2 Logistic Regression

A regressão logística multinomial, também conhecida como *Softmax Regression*, devido à função de hipótese que utiliza, é um algoritmo de aprendizado supervisionado que pode ser utilizado em diversos problemas incluindo classificação de texto. É um modelo de regressão que generaliza a regressão logística para problemas de classificação onde a saída pode assumir mais de dois valores possíveis.

Considere que $D = \{(x^n, t^n)\}_{n=1}^{l}$ representa a amostra de treinamento, onde $x^n \in X \subset R^d$ é o vetor de características de entrada para o *i-ésimo* exemplo e $t^n \in T = \{t | t \in \{0,1\}, ||t||_1 = 1\}$ é o vetor correspondente de saídas desejadas, usando o esquema usual de codificação *1-de-m*. A regressão logística multinomial constrói um modelo linear generalizado com uma função inversa *softmax*, permitindo que as saídas sejam interpretadas como estimativas a posteriori das probabilidades pertencentes à classe [Cawley, Talbot e Girolami 2007],

$$p(t_i{}^n | x^n) = y_i{}^n = \frac{exp\{a_i{}^n\}}{\sum_{j=1}^{c} exp\{a_j^n\}} \tag{2-8}$$

onde

$$a_i{}^n = \sum_{j=1}^{d} w_{ij} x_j^n \tag{2-9}$$

Assumindo que D representa uma i.i.d. (distribuição idêntica e independente) amostra de uma distribuição multinomial condicional. Então o *log* de verossimilhança negativo, usado como uma medida do desajuste de dados, pode ser escrito como [Cawley, Talbot e Girolami 2007],

$$E_D = \sum_{n=1}^{l} E_D{}^n = -\sum_{n=1}^{l}\sum_{i=1}^{c} t_i^n log\{y_i^n\} \tag{2-10}$$

Os parâmetros w do modelo de regressão logística multinomial são dados pelo minimizador de um critério de treinamento de máxima verossimilhança penalizado [Cawley, Talbot e Girolami 2007],

$$L = E_D + \alpha E_w \tag{2-11}$$

onde

$$E_w = \sum_{n=1}^{l}\sum_{i=1}^{c} |w_{ij}| \tag{2-12}$$

Segundo [Genkin, Lewis e Madigan 2007], para um problema de categorização de documentos, $p(y = +1|x_i)$ será uma estimativa da probabilidade do *i-esimo* documento pertencer à categoria. A decisão de atribuir a categoria pode ser com base na comparação da estimativa de probabilidade com um limite ou, de forma mais geral, computando qual decisão dá o melhor resultado esperado. Para um modelo de regressão logística fazer predições precisas para documentos ainda não conhecidos, devemos evitar o sobreajuste (*overfitting*) dos dados de treinamento.

2.2.2.3 SVM

A máquina de vetores de suporte (SVM), descrita pela primeira vez por [Boser, Guyon e Vapnik 1992], em 1992, rapidamente se estabeleceu como uma abordagem algorítmica poderosa para o problema de classificação dentro do contexto mais amplo conhecido como aprendizado supervisionado [Press et al. 1992]. Muitos problemas de classificação cujas soluções eram anteriormente dominadas por redes neurais e métodos mais complicados podem ser resolvidos de forma direta por SVM [Cristianini, Shawe-Taylor et al. 2000]. De acordo com [Press et al. 1992], no problema de classificação de aprendizado supervisionado, recebemos um conjunto de dados de treinamento que consiste em m pontos

$$(x_i, y_i) \qquad i = 1, ..., m \tag{2-13}$$

Cada x_i é um vetor de características em n dimensões que descreve o ponto de dados, enquanto cada y_i correspondente tem o valor ± 1, indicando se esse ponto de dados está dentro (+1) ou fora (-1) do conjunto que queremos que aprenda a reconhecer. Precisamos de uma regra de decisão, na forma de uma função $f(x)$, cujo sinal prediz o valor de y. Não apenas para os dados no conjunto de treinamento, mas também para novos valores de x nunca vistos antes.

Pode-se entender o SVM, conceitualmente, como uma série de generalizações a partir de um ponto de partida idealizado. O ponto de partida é o caso especial de dados linearmente separáveis [Press et al. 1992]. Neste caso, somos informados que existe um hiperplano em n dimensões, ou seja, uma superfície n - *1* dimensional definida pela equação 2-14, que separa completamente os dados de treinamento.

$$f(x) \equiv w.x + b = 0 \tag{2-14}$$

Em outras palavras, todos os pontos de treinamento com $y_i = 1$ estão de um lado do hiperplano, e portanto têm $f(x) > 0$, enquanto todos os pontos de treinamento com $y_i = -1$ estão do outro lado, e têm $f(x) < 0$. Tudo que é preciso fazer é encontrar w (um vetor normal para o hiperplano) e b (um deslocamento). Então $f(x)$ na equação 2-14 será a regra de decisão.

Em geral, mais de um hiperplano separará dados linearmente separáveis. Vamos escolher o hiperplano que tem a maior margem, ou seja, maximiza a distância perpendicular aos pontos mais próximos do hiperplano em ambos os lados. Especificamente, dado qualquer hiperplano que separa os dados, podemos sempre dimensionar w por uma constante e ajustar b apropriadamente, para fazer

$$\begin{aligned} w.x + b &\geq 0 \qquad quando \;\; y_i = +1 \\ w.x + b &\leq 0 \qquad quando \;\; y_i = -1 \end{aligned} \tag{2-15}$$

Essas equações representam hiperplanos delimitadores paralelos que separam os dados fazendo a classificação dos documentos.

2.2.2.4 Naive Bayes

Segundo [Manning, Raghavan e Schütze 2010], Multinomial Naive Bayes é um método de aprendizagem probabilístico. A probabilidade de um documento d pertencer à classe c é calculada como

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{2-16}$$

Onde $P(t_k|c)$ é a probabilidade condicional do termo t_k ocorrer em um documento de uma determinada classe c. $P(t_k|c)$ é interpretado como uma medida de quanta evidência t_k contribui para que c seja a classe correta. $P(c)$ é a probabilidade anterior de um documento ocorrer na classe c. Se os termos de um documento não fornecem evidências claras para uma classe em relação a outra, escolhemos aquele que tem uma probabilidade anterior mais alta. $\langle t_1, t_2, ..., t_{n_d} \rangle$ são as características (*tokens*) em d que fazem parte do vocabulário que usamos para a classificação e n_d é o número de características (*tokens*) em d.

Na classificação de documentos, o objetivo é encontrar a melhor classe para o documento. A melhor classe na classificação do algoritmo Naive Bayes é o mapa de classe c mais provável ou máximo a posteriori (MAP) c_{map}:

$$c_{map} = argmax \hat{P}(c|d) = argmax \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c) \tag{2-17}$$

Escreve-se $\hat{P}$ para P porque não é possível saber os verdadeiros valores dos parâmetros $P(c)$ e $P(t_k|c)$, mas são estimados do conjunto de treinamento.

[Manning, Raghavan e Schütze 2010] definem que muitas probabilidades condicionais são multiplicadas, uma para cada posição $1 \leq k \leq n_d$. Isso pode resultar em um estouro negativo de ponto flutuante. Portanto, é melhor realizar o cálculo adicionando logaritmos de probabilidades em vez de multiplicar probabilidades. A classe com a pontuação de probabilidade de *log* mais alta é a mais provável; *log (xy) = log (x) + log (y)* e a função logaritmo é monotônica. Portanto, a maximização que realmente é feita na maioria das implementações do Naive Bayes é

$$c_{map} = argmax[log\ \hat{P}(c) + \sum_{1 \leq k \leq n_d} log\ \hat{P}(t_k|c)] \tag{2-18}$$

Cada parâmetro condicional $log \hat{P}(t_k|c)$ é um peso que indica quão bom um indicador t_k é para c. Da mesma forma, o $log\ \hat{P}(c)$ anterior é um peso que indica a frequência relativa de c. As classes mais frequentes têm maior probabilidade de ser a classe correta do que as classes não frequentes. A soma do *log* anterior e dos pesos dos termos é então uma medida de quanta evidência existe para o documento pertencer a uma classe, e a equação 2-18 seleciona a classe para a qual tem mais evidências.

2.2.2.5 Random Forest

[Breiman 2001] propõe uma variante de amostragem (*bagging*) que chama de floresta aleatória. Floresta aleatória (*RF - Random Forest*) é uma classe geral de métodos de construção de conjuntos usando uma árvore de decisão como classificador base. Para ser chamado de floresta aleatória, um conjunto de árvores de decisão deve ser construído gerando vetores aleatórios independentes distribuídos de forma idêntica $\theta_k, k = 1, ..., L$, usando cada vetor para aumentar a árvore de decisão.

Conforme definição de [Breiman 2001], uma floresta aleatória é um classificador que consiste em uma coleção de classificadores estruturados em árvore $\{h(\mathbf{x}, \theta_k), k = 1, ...\}$, onde $\{\theta_k\}$ são vetores aleatórios distribuídos de forma idêntica e independentemente, e cada árvore lança um voto unitário para a classe mais popular na entrada $\mathbf{x}$.

Assim, uma floresta aleatória pode ser construída por amostragem do conjunto de características, do conjunto de dados ou apenas variando aleatoriamente alguns dos parâmetros da árvore. Qualquer combinação dessas fontes de diversidade também levará a uma floresta aleatória. Por exemplo, podemos obter amostras do conjunto de características e também do conjunto de dados. Desta forma, o vetor aleatório $\{\theta_k\}$ será uma junção de duas amostras iniciais, uma do conjunto de treinamento de pontos de dados e uma do conjunto de características, contendo $N + n$ elementos ao todo [Kuncheva 2014].

Uma forma de implementação das florestas aleatórias é a seleção aleatória de características (entradas) [Kuncheva 2014]. A seleção aleatória de características é realizada em cada subárvore da floresta. É escolhido aleatoriamente um subconjunto S, com M características, do conjunto original de n características. Cada subárvore da floresta recebe, como entrada, um subconjunto S. Para ajustar o modelo é possível utilizar validação cruzada para fornecer novos subconjuntos de características para cada subárvore até encontrar o modelo ideal. O valor recomendado de M é $log_2 n$, onde n é o número total de características [Breiman 2001].

De acordo com [Fernández-Delgado et al. 2014], florestas aleatórias são consideradas um dos classificadores de maior sucesso na tarefa de classificação automática. Essa alta eficiência deve-se ao grande conjunto de árvores de baixa correlação que compõem a floresta. Este resultado é obtido escolhendo os dados de treinamento com procedimentos aleatórios, como *bagging* ou seleção aleatória de características. Outra abordagem que melhora a precisão das florestas aleatórias é aumentar a profundidade máxima das árvores. Ao considerar a média de várias árvores correlacionadas como a previsão do modelo, pode-se mostrar que o classificador RF reduz a variância, mantendo seu viés inalterado [Breiman 2001].

2.2.2.6 kNN

A busca dos k vizinhos mais próximos (k-Nearest Neighbour - kNN) é um algoritmo para encontrar os k pontos mais próximos no espaço de características. Formalmente, o problema de pesquisa kNN é definido da seguinte maneira: dado um conjunto $X = \{x_1, x_2, \ldots, x_n\}$ de n pontos no espaço $\mathbb{R}^d$ e um ponto de consulta q também em $\mathbb{R}^d$, encontre os k pontos mais próximos a q no conjunto X. A proximidade depende da medida de distância utilizada, que pode ser a distância euclidiana, a distância do cosseno ou qualquer outra distância que seja adequada ao contexto [Amorin et al. 2018].

O classificador kNN apesar de ser um algoritmo de alto custo computacional é muito conhecido e amplamente utilizado na tarefa de classificação em aprendizado de máquina [Kuang e Zhao 2009]. O método kNN primeiro seleciona as amostras de treinamento mais próximas para uma amostra de teste e, em seguida, faz a predição da amostra de teste com a classe majoritária entre as amostras de treinamento que compõem a vizinhança da instância de teste [Deng et al. 2016]. No entanto, o kNN precisa calcular a distância (ou similaridade) de todas as amostras de treinamento para cada amostra de teste no processo de seleção de vizinhos. A vizinhança para cada amostra de teste x (ainda não conhecida para o algoritmo kNN) é definida pelo conjunto de k instâncias mais próximas de x no espaço de entrada (amostras de treinamento), onde a proximidade implica em uma métrica, como distância euclidiana ou similaridade de cosseno. Após ordenar os resultados do cálculo das distâncias, a decisão do rótulo da classe da amostra de teste x pode ser feita de acordo com o rótulo das k amostras mais próximas no conjunto de amostras de treinamento. Baseado na hipótese de contiguidade, é esperado que uma amostra de teste x tenha a mesma classificação que as amostras de treinamento localizadas na vizinhança de x [Manning, Raghavan e Schütze 2010].

A qualidade do conjunto de dados do treinamento afeta diretamente o resultado da classificação. Dessa forma, a escolha do parâmetro k também é muito importante, pois diferentes valores de k podem resultar em rótulos de classificação diferentes para a mesma amostra de teste x.

Uma maneira de resolver o problema kNN é usar uma abordagem de força bruta para encontrar os vizinhos mais próximos de um ponto. Consiste em computar todas as distâncias entre q e os pontos em X, ordenando as distâncias calculadas e selecionando os k pontos correspondentes às k menores distâncias. No entanto, o cálculo real das distâncias e a seleção dos vizinhos mais próximos para grandes conjuntos de dados requerem alto custo de computação, $\mathcal{O}(nd)$ para calcular as n distâncias e $\mathcal{O}(n \log n)$ para classificá-los.

2.2.3 Ensembles

Conjunto (*ensemble*) de classificadores é uma técnica de aprendizado de máquina amplamente estudada [Rokach 2009]. Entre vários métodos de aprendizado de máquina usados na prática, conjunto (*ensemble*) é uma das mais efetivas em várias aplicações. Segundo [Woźniak, Graña e Corchado 2014], conjuntos de classificadores homogêneos são compostos de classificadores do mesmo tipo, enquanto conjuntos de classificadores heterogêneos combinam diversos tipos de classificadores. O primeiro grupo constrói o modelo usando o mesmo classificador base e depende do aprendizado de muitas versões da mesma técnica de classificação, cada uma construída com alguma forma de alteração no conjunto de treinamento. Então, geralmente uma média das predições é calculada para chegar à decisão final com capacidade de generalização mais alta. O segundo grupo constrói o modelo com classificadores base diferentes. Combina um conjunto de métodos de aprendizagem distintos, treinados com o mesmo conjunto de dados de treinamento e em seguida é produzida a decisão final de acordo com as predições realizadas pelos classificadores do conjunto.

A construção de conjuntos de classificadores homogêneos ou heterogêneos pode ser feita de várias formas. Considerando a classificação de documentos, [Dong e Han 2004] usam o que eles chamam de Moderated Asymmetric Naïve Bayes (MANB) como um aprendiz base em seu conjunto de classificadores homogêneo. [Campos et al. 2017] combinam duas técnicas bem conhecidas de conjuntos homogêneos, *bagging* e *boosting*, explorando *Random Forest* como um "aprendiz fraco" (*weak learner*) no *boosting framework*. A combinação é alcançada por meio da atualização dos pesos apenas das amostras que não participaram do treinamento (*out-of-bag*), conforme definido em [Breiman 2001]. Essa combinação mitiga o *overfitting* apresentado por modelos *Random Forest* em tarefas de classificação de documentos e aumenta seu poder de generalização.

Existem duas técnicas comuns para combinar predições de diferentes classificadores, chamadas de métodos de combinação fixa e métodos de combinação treináveis [Nguyen et al. 2016]. Uma vantagem de aplicar métodos fixos para conjuntos (*ensembles*) de métodos de aprendizagem é que não há a necessidade de treinar um meta-classificador desde que eles não levam em consideração o meta-nível de conjunto de treinamento quando combinam os aprendizes (*learners*). Assim, eles consomem menos tempo de computação que outros métodos. Vários métodos de combinação fixa podem ser encontrados na literatura, como *Sum*, *Product*, *Vote* e *Average* [Bauer e Kohavi 1999] e [Kuncheva 2014].

De outra forma, os métodos de combinação treináveis funcionam em meta-nível de dados, com o intuito de aprender como combinar os resultados dos aprendizes (*le-*

arners) base. Embora a exploração de meta-nível de dados, para extrair conhecimento, geralmente aproveite a eficácia da classificação, isso vem com o custo de esforço computacional adicional [Nguyen et al. 2016].

O presente trabalho constrói (implementa) um *ensemble* heterogêneo, com a utilização de seis métodos de aprendizagem diferentes para chegar à predição final. Em nosso *ensemble* são utilizados os seguintes métodos de aprendizagem:

- *SGD*
- *Logistic Regression*
- *SVM*
- *Naive Bayes*
- *Random Forest*
- *kNN*

2.2.4 Testes estatísticos

Nos últimos anos, a comunidade de aprendizado de máquina tem se tornado cada vez mais consciente da necessidade de validação estatística dos resultados publicados. Com o desenvolvimento de novos algoritmos ou a modificação de algoritmos existentes é possível fazer a comparação entre estes métodos para avaliar seus desempenhos [Demšar 2006]. Várias soluções para este problema foram propostas e o objetivo é distinguir o modelo de aprendizagem bem-sucedido do mal-sucedido. Segundo [Demšar 2006], vários conjuntos de dados de teste são selecionados para teste, os algoritmos são executados e a qualidade dos modelos resultantes é avaliada usando uma métrica apropriada, mais comumente a precisão da classificação. Dessa forma é possível verificar estatisticamente a hipótese de melhoria de desempenho.

De acordo com [Demšar 2006], vários pesquisadores abordaram o problema de comparar dois classificadores em um único conjunto de dados e propuseram várias soluções. Como exemplo, podem ser cidtados os sequintes testes: *t*-testes pareados, teste de McNemar e validação cruzada 5x2. O presente trabalho abordará apenas as técnicas de *t*-testes pareados.

Conforme definido em [Dietterich 1998], considere a tarefa de ADC, um conjunto de dados D e N classes. Uma função f, que classifica cada documento $d \in D$ em uma das N classes. Retire aleatoriamente uma amostra S de D de acordo com uma distribuição de probabilidade P. Um conjunto de dados de treinamento é construído rotulando cada $s \in S$ de acordo com $f(s)$. Cada documento de treinamento tem a forma $\langle s, f(s) \rangle$. Em algumas aplicações, pode haver uma fonte de ruído de classificação que define aleatoriamente o rótulo com um valor incorreto.

De acordo com [Dietterich 1998], um algoritmo de aprendizado A recebe como entrada um conjunto de dados de treinamento R e gera um classificador $\hat{f}$. A verdadeira taxa de erro desse classificador é a probabilidade de que $\hat{f}$ classifique incorretamente um exemplo tirado aleatoriamente de D de acordo com P. Na prática, essa taxa de erro é estimada tomando a amostra disponível S e subdividindo-a em um conjunto de treinamento R e um conjunto de teste T. A taxa de erro de $\hat{f}$ em T fornece uma estimativa da taxa de erro verdadeira de $\hat{f}$ no conjunto D [Dietterich 1998].

A hipótese nula a ser testada é que para um conjunto de treinamento R aleatório, de tamanho fixo, os dois algoritmos de aprendizagem terão a mesma taxa de erro em um exemplo de teste extraído aleatoriamente de D, onde todas as escolhas aleatórias são feitas de acordo com a distribuição P. Seja $\hat{f_A}$ a saída do classificador pelo algoritmo A, treinado no conjunto de treinamento R, e seja $\hat{f_B}$ a saída do classificador pelo algoritmo B treinado em R [Dietterich 1998].

2.2.4.1 *t*-teste pareado reamostrado

Segundo [Dietterich 1998], este teste estatístico é atualmente o mais popular na literatura de aprendizado de máquina. Uma série de i tentativas é conduzida. Considere dois algoritmos de aprendizagem A e B e um conjunto de dados S. O conjunto de dados é dividido em conjunto de treinamento R e conjunto de teste T, geralmente *2/3* e *1/3*. Os algoritmos de aprendizagem A e B são treinados em R e os classificadores resultantes são testados em T. Seja $p_A^{(i)}$ e $p_B^{(i)}$ as proporções observadas de exemplos de teste classificados incorretamente pelos algoritmos A e B, durante a tentativa i. Se for assumido que as i diferenças de $p^{(i)} = p_A^{(i)} - p_B^{(i)}$ foram construídas independentemente de uma distribuição normal, então é aplicado o t-teste *Student*, para calcular a estatística

$$t = \frac{\bar{p}.\sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n}(p^{(i)} - \bar{p})^2}{n-1}}} \tag{2-19}$$

onde $\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p^{(i)}$. Sob a hipótese nula, essa estatística tem uma distribuição t com $n - 1$ graus de liberdade. Para *10* tentativas, a hipótese nula pode ser rejeitada se $|t| > t_{9,0.975} = 2.262$ [Dietterich 1998].

2.2.4.2 *t*-teste pareado com validação cruzada em *k-folds*

Este teste é idêntico ao t-teste pareado, exceto que, em vez de construir cada par de conjuntos de treinamento e teste dividindo S aleatoriamente, o conjunto de dados S é dividio aleatoriamente em k conjuntos disjuntos de tamanhos iguais, $T_1, \ldots, T_k$. Em seguida, são realizadas k tentativas de teste. Em cada tentativa, o conjunto de teste é T_i,

e o conjunto de treinamento é a união de todos os outros $T_j; j \neq i$. É realizado o mesmo cálculo estatístico do *t*-teste pareado [Dietterich 1998]. A vantagem dessa abordagem é que cada conjunto de teste é independente dos outros

CAPÍTULO 3

Seleção customizada de método de classificação

3.1 Descrição do problema e trabalhos relacionados

A classificação é uma tarefa fundamental no reconhecimento de padrões, razão pela qual nas últimas décadas houve um grande número de projetos de pesquisa dedicados a métodos de classificação aplicados a diferentes campos da atividade humana [Jr, Sabourin e Oliveira 2014]. Diante da grande aplicabilidade da tarefa de classificação, foram propostos vários trabalhos na literatura que, apesar de serem diferentes em muitos aspectos, demonstram a inviabilidade da criação de um classificador único (monolítico) capaz de cobrir todas as variâncias decorrentes dos problemas de classificação [Jr, Sabourin e Oliveira 2014]. Baseadas neste cenário, várias pesquisas recentes concentraram-se na criação de Sistemas de Múltiplos Classificadores (MCS) para criar abordagens que sejam capazes de tratar a grande variabilidade nos problemas de classificação.

Referindo-se a problemas de classificação, o teorema de [Wolpert 2002] especifica que não há uma única abordagem de modelagem de classificador que seja ideal para todas as tarefas de reconhecimento de padrões, uma vez que cada um tem seu próprio domínio de competência [Woźniak, Graña e Corchado 2014]. Ainda segundo [Woźniak, Graña e Corchado 2014], para uma dada tarefa de classificação, esperamos que o MCS explore os pontos fortes dos modelos de classificadores individuais à nossa disposição para produzir sistemas de reconhecimento compostos, de alta qualidade, que superem o desempenho dos classificadores individuais. Recentemente, a pesquisa em reconhecimento de padrões, tem focado na integração de vários sistemas classificadores, que podem ser construídos seguindo as abordagens de conjunto de classificadores homogêneos (utilizam os mesmos classificadores base na construção do modelo) ou heterogêneos (utilizam classificadores base diferentes na construção do modelo) e/ou abordagens diferentes de construção de conjuntos de dados. Esses sistemas realizam a integração de informações de decisões de classificação em diferentes níveis, superando as limitações das abordagens tradicionais baseadas em classificadores únicos [Woźniak, Graña e Corchado 2014].

Sistemas de Múltiplos Classificadores (MCS) é uma área de pesquisa

amplamente ativa em aprendizado de máquina e reconhecimento de padrões [Cruz, Sabourin e Cavalcanti 2018]. Nos últimos anos, vários estudos foram publicados demonstrando suas vantagens sobre os modelos de classificadores individuais baseados em avaliações teóricas [Kuncheva 2002] e [Kuncheva 2014] e empíricas [Fernández-Delgado et al. 2014] e [Polikar 2006]. Estas técnicas são amplamente usadas para resolver vários problemas do mundo real, como classificação de gênero musical [Almeida et al. 2012], sistemas de recomendação [Jahrer, Töscher e Legenstein 2010] e reconhecimento de padrões [Cruz, Cavalcanti e Ren 2011], dentre outros. Estes sistemas são compostos por classificadores base, como por exemplo, *SGD*, *Logistic Regression*, *SVM*, *kNN* e *Random Forest*, que formam um conjunto de classificadores (*ensemble*), que serão integrados para dar a decisão final sobre a classificação dos documentos de consulta.

Segundo definição em [Cruz, Sabourin e Cavalcanti 2018], os Sistemas de Múltiplos Classificadores (exemplo apresentado na figura 3.1), geralmente são compostos pelas três etapas a seguir:

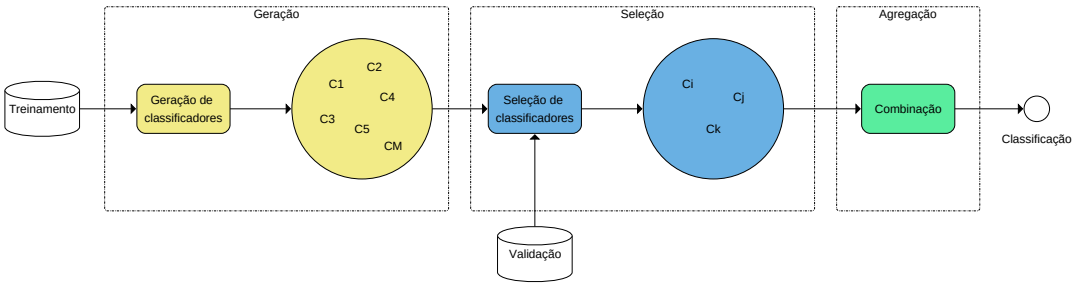
1. **Geração dos classificadores**: o objetivo desta etapa é definir um conjunto de classificadores base, $C = C_1, ..., C_M$, contendo M classificadores que são precisos e diversificados. Os classificadores básicos devem ser diferentes, pois não há razão para combinar classificadores que sempre apresentam a mesma saída. Existem seis estratégias principais para gerar um conjunto diversificado de classificadores [Kuncheva 2014] e [Duin 2002].
2. **Seleção do(s) classificador(es)**: nesta etapa um classificador ou um subconjunto com os melhores classificadores é selecionado. Existem duas abordagens de seleção: estática e dinâmica. No método de seleção estático o subconjunto de classificadores é selecionado durante a fase de treinamento de acordo com um critério de seleção (geralmente diversificação e precisão) baseado no conjunto de dados de validação. O mesmo subconjunto de classificadores é usado para fazer a predição de todos os documentos de consulta. Na seleção dinâmica, um classificador simples ou um subconjunto de classificadores é selecionado para classificar cada amostra de documento de consulta. Esta técnica consiste em encontrar o classificador ou subconjunto de classificadores mais competente para cada documento de consulta $\mathbf{x}_i$. Cada classificador base é um especialista em regiões distintas do espaço de características e esta técnica objetiva selecionar o classificador mais competente na região local onde $\mathbf{x}_i$ está localizado.
3. **Agregação ou combinação**: esta etapa consiste em agregar (combinar) as saídas (predições) feitas pelos classificadores selecionados de acordo com uma regra de combinação. Segundo [Cruz, Sabourin e Cavalcanti 2018], existem três estratégias principais para a fase de agregação: não treinável (as saídas dos classificadores são

avaliadas diretamente), treinável (nesta estratégia as saídas dos classificadores base são usadas como recursos de entrada para outro algoritmo de aprendizagem, que aprende a função de agregação com base nos dados de treinamento) e ponderação dinâmica (as saídas dos classificadores são agregadas para dar a decisão final de forma que o classificador mais competente receba um valor de peso mais alto e assim por diante).

Um MCS pode ser construído utilizando uma das várias abordagens descritas na literatura, os trabalhos [Kuncheva 2014], [Polikar 2006], [Woźniak, Graña e Corchado 2014] e [Ren, Zhang e Suganthan 2016] apresentam revisões que definem diferentes aspectos do MCS, como por exemplo:

- Como interconectar classificadores individuais;
- A forma de conduzir a geração e seleção de um conjunto de classificadores;
- Como construir uma função de agregação de decisão (predição final) que pode explorar os pontos fortes dos classificadores selecionados e combiná-los de forma otimizada.

Figura 3.1: Visão geral de um Sistema de Múltiplos Classificadores (figura adaptada de [Cruz, Sabourin e Cavalcanti 2018]).

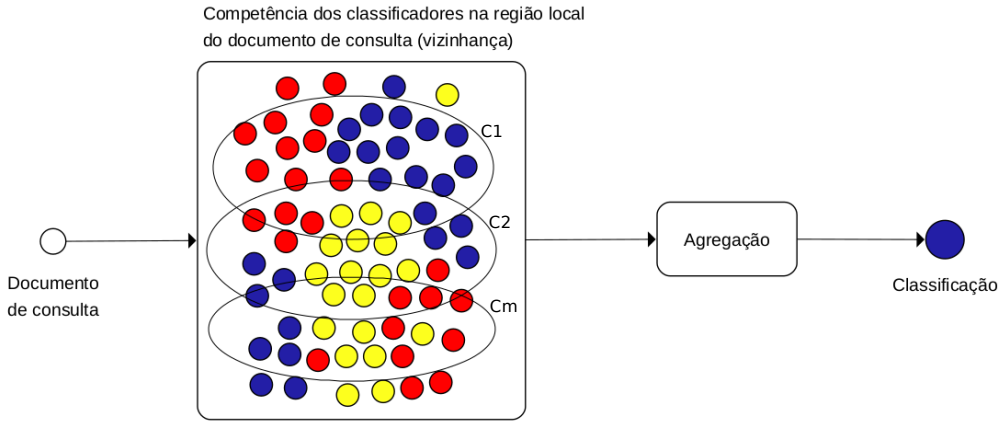


Uma das abordagens de MCS mais promissoras é a Seleção Dinâmica (DS), nesta abordagem os classificadores base são selecionados em tempo real à medida que o MCS recebe cada nova amostra de documento de consulta para ser classificada. A seleção dinâmica (DS) do classificador tornou-se uma pesquisa importante nos últimos anos [Cruz, Sabourin e Cavalcanti 2018]. As técnicas de DS avaliam o nível de competência de cada classificador, do conjunto de classificadores, na região de competência, que é definida como a região local em torno do documento de consulta. O nível de competência do classificador base é estimada nesta região local [Cruz, Sabourin e Cavalcanti 2018]. Apenas o classificador mais competente, ou o subconjunto dos classificadores mais competentes, é selecionado para classificar o documento de consulta.

Na seleção dinâmica, a etapa principal é definir um critério para medir o nível de competência do classificador base, ou seja, saber como selecionar o(s) classificador(es) mais competente(s) para cada um dos documento de consulta.

Normalmente, a competência dos classificadores é estimada com base em uma região local do espaço de características (termos) onde o documento de consulta está localizado, conhecida como região de competência, [Cruz, Sabourin e Cavalcanti 2018], [Ren, Zhang e Suganthan 2016], [Giacinto e Roli 2001] e [Woods, Kegelmeyer e Bowyer 1997]. Esta região pode ser definida por diferentes métodos, como a aplicação do *k-NN* (k vizinhos mais próximos) [Jr, Sabourin e Oliveira 2014] e [Woods, Kegelmeyer e Bowyer 1997], usando técnicas de agrupamento [Kuncheva 2000] ou com a utilização do espaço de decisão (Espaço de Conhecimento Comportamental - BKS [Huang e Suen 1995] e [Giacinto e Roli 2001]). Em seguida, estima-se o nível de competência dos classificadores base considerando apenas os documentos pertencentes a essa região local e de acordo com algum critério de seleção como a precisão dos classificadores nesta região local [Woods, Kegelmeyer e Bowyer 1997] ou ranqueamento [Sabourin et al. 1993] e modelos probabilísticos [Woloszynski et al. 2012]. O(s) classificador(es) que alcançaram um determinado nível de competência são selecionados para realizar a classificação do(s) documento(s) de consulta. A figura 3.2 apresenta uma visão geral da seleção dinâmica do classificador, baseada na região local de competência (vizinhança do documento de consulta).

Figura 3.2: Seleção dinâmica do classificador, baseada na nível de competência do classificador na região local do documento de consulta.



3.1.1 Descrição do problema

Considere a tarefa de classificação automática de documentos para N classes $W_1, ..., W_N$. Cada documento é representado por um vetor de caraceterísticas $\mathbf{X}$ e $R(\mathbf{X})$

representa seu rótulo (classe a que pertence). Assuma que M classificadores diferentes C_i, $i = \{1, ..., M\}$ (M é o tamanho do conjunto de classificadores), foram treinados para resolver o problema de Classificação Automática de Documentos, considere que $C_i(\mathbf{X}) \in \{1, ..., M\}$ indica o rótulo (classe) associado ao documento $\mathbf{X}$ pelo classificador C_i (um classificador base pertencente ao conjunto C). Então para cada documento de consulta, ainda não conhecido pelo conjunto de classificadores, é realizada a seleção dinâmica do classificador (DCS) ou a seleção dinâmica do subconjunto de classificadores (DES) de M, que tem o melhor nível de competência na região local (LA) para classificar o documento de consulta X [Giacinto e Roli 2001].

Para estimar o nível de competência de um classificador base C_i para um documento de consulta X_i é calculada a precisão local do classificador utilizando a abordagem k-NN, que define a região local de competência. Neste caso, os k vizinhos mais próximos do documento de consulta X_i são encontrados e o conjunto desses documentos, chamado de região de competência, é definido por $\theta = \{x_i, ..., x_k\}$. Em seguida, a competência dos classificadores base é estimada levando em consideração apenas as instâncias pertencentes a esta região local de competência. Após selecionar o subconjunto de classificadores mais competentes é feita a agregação dos resultados (predições) e a classificação do documento de consulta X_i.

A competência do classificador define o quanto pode-se confiar em um classificador especialista, dada uma tarefa de classificação. A noção de competência utilizada é extensiva ao campo do aprendizado de máquina como forma de selecionar, entre uma grande quantidade de diferentes modelos de classificação, aquele que melhor se ajusta ao problema em questão [Cruz et al. 2015].

3.1.2 Trabalhos relacionados

Referindo-se a problemas de classificação, o teorema de Wolpert define que não há uma abordagem de modelagem de classificador único que seja ideal para todas as tarefas de reconhecimento de padrões, uma vez que cada um tem seu próprio domínio de competência. Para uma dada tarefa de classificação, é esperado que o MCS explore os pontos fortes dos modelos de classificadores individuais disponíveis para produzir o sistema de reconhecimento com um conjunto de classificadores de alta qualidade, superando o desempenho dos classificadores individuais. [Wolpert 2002]

De maneira específica [Polikar 2006] analisa as condições sob as quais sistemas de múltiplos classificadores podem ser melhores do que a abordagem de classificador único. Também analisa algoritmos para gerar os classificadores individuais dos sistemas de múltiplos classificadores, e vários procedimentos através dos quais os classificadores individuais podem ser combinados. São discutidos algoritmos baseados em conjuntos conhecidos, como *Bagging*, *Boosting*, *AdaBoost*, empilhamento e mistura hierárquica

de especialistas; bem como regras de combinação de classificadores comumente usadas, incluindo combinação algébrica de resultados (predições) geradas pelos classificadores, técnicas baseadas em votação, espaço de conhecimento comportamental (BKS) e modelos de decisão. Além disso, discutem sobre as pesquisas atuais e futuras a respeito de novas aplicações dos sistemas de múltiplos classificadores.

Um foco atual de intensa pesquisa em classificação de padrões é a combinação de vários sistemas classificadores, que podem ser construídos seguindo modelos iguais (homogêneos) ou diferentes (heterogêneos). Esses sistemas realizam a fusão de informações de decisões de classificação em diferentes níveis, superando as limitações das abordagens tradicionais baseadas em classificadores únicos. O trabalho [Woźniak, Graña e Corchado 2014] apresenta um levantamento atualizado sobre sistemas de múltiplos classificadores (MCS) do ponto de vista de Sistemas Híbridos Inteligentes. De acordo com os autores os classificadores que implementam o processo de decisão inteligente também estão sujeitos à hibridização através de várias formas de combinação. O trabalho também discute questões importantes, como diversidade dos classificadores e métodos de combinação de decisão entre os classificadores, além de fornecer uma visão dos tipos de aplicações que estão sendo desenvolvidas atualmente com a abordagem MCS.

[Jr, Sabourin e Oliveira 2014] apresenta uma revisão da literatura sobre sistemas de múltiplos classificadores (MCS) baseados na seleção dinâmica de classificadores. É apresentado o estado da arte em MCS organizado de acordo com uma taxonomia proposta pelos autores. Além disso, uma análise em duas etapas é aplicada aos resultados dos principais métodos relatados na literatura, considerando diferentes problemas de classificação. A primeira etapa é baseada em análises estatísticas da significância desses resultados. A ideia é descobrir os problemas para os quais uma contribuição significativa pode ser observada em termos de desempenho de classificação, usando uma abordagem de seleção dinâmica. A segunda etapa, baseada em medidas de complexidade de dados, é usada para investigar se existe ou não uma relação entre a possível contribuição de desempenho e a complexidade do problema de classificação. Os autores concluem que, para alguns problemas de classificação, a seleção dinâmica é estatisticamente significativa quando comparada com a de um classificador base único.

No artigo [Cavalin, Sabourin e Suen 2013] é proposta uma nova abordagem para a seleção dinâmica no conjunto de classificadores. Baseados no conceito organizações multiestágios, cujo objetivo principal é definir uma função de fusão multicamadas adaptada a cada problema de reconhecimento de padrões, os autores propõem a organização dinâmica multiestágios (*DMO*), que define a melhor estrutura de vários estágios para cada documento de consulta (teste) a ser classificador. Os autores estendem a abordagem de [Santos, Sabourin e Maupin 2008] e propõem as implementações *DSA m* e *DSA c* para

DMO. A primeira abordagem considera um conjunto de funções de seleção dinâmica para generalizar uma estrutura *DMO*. A segunda abordagem considera informações contextuais, representadas pelas saídas obtidas pelos classificadores no conjunto de dados de validação. A avaliação dos experimentos, demonstrou que o *DSA c* teve melhores resultados que o *DSA m* na maioria dos problemas, mostrando que o uso de informações contextuais pode atingir um desempenho melhor do que outros métodos existentes. Apoiados por experimentos, também observaram que a seleção dinâmica é geralmente preferida em vez de abordagens estáticas, quando o problema de reconhecimento apresenta um alto nível de incerteza.

No trabalho [Cruz, Sabourin e Cavalcanti 2018], os autores apresentam os avanços recentes em Sistemas de Múltiplos Classificadores (MCS) no reconhecimento de padrões (classificação). É datalhada a técnica Seleção Dinâmica (DS), em que os classificadores base são selecionados em tempo real, de acordo com cada novo documento de consulta a ser classificada. Este trabalho apresenta uma revisão das técnicas de DS propostas na literatura do ponto de vista teórico e empírico. Define as principais características encontradas na Seleção Dinâmica: (1) A metodologia usada para definir uma região local para a estimativa da competência local dos classificadores base; (2) A fonte de informação usada para estimar o nível de competência dos classificadores de base, como precisão local, ranqueamento e modelos probabilísticos, e (3) A abordagem de seleção, que determina se um único classifador ou um conjunto de classificadores é selecionado. Além disso, também apresentam várias perspectivas e questões de pesquisa que podem ser usadas como um guia para trabalhos futuros neste domínio.

A criação de um MCS depende da escolha dos classificadores base que irão compor o conjunto de classificadores. Existem técnicas que auxiliam nessa escolha. A diversidade entre os membros de um conjunto de classificadores é considerada uma questão chave na combinação de classificadores e pode ser usada para escolher os classificadores base. No trabalho [Kuncheva e Whitaker 2003] são estudadas dez estatísticas que podem medir a diversidade entre as saídas do classificador binário (voto correto ou incorreto para o rótulo da classe): quatro medidas pareadas médias (a estatística Q, a correlação, a discordância e a falha dupla) e seis medidas não pareadas (a entropia dos votos, o índice de dificuldade, a variância de Kohavi-Wolpert, a concordância entre avaliadores, a diversidade generalizada e a diversidade de falha coincidente). Os autores também apresentam experimentos que examinam a relação entre a precisão do conjunto de classificadores e as medidas de diversidade.

[Duin 2002] apresenta um estudo sobre formas de definir um conjunto de classificadores. O autor defende que classificadores base devem ser diferentes (já que não faz sentido combinar classificadores idênticos), mas eles também devem ser comparáveis, ou seja, suas saídas devem ser representadas de forma que possam ser combinadas em

uma decisão final. O autor define que um conjunto consistente de classificadores diferentes pode ser gerado utilizando: inicializações diferentes, escolha de parâmetros diferentes (redes neurais), classificadores com arquiteturas diferentes (SVM, k-NN), conjuntos de treinamento diferentes e conjuntos de características (representação dos documentos) diferentes. Neste trabalho também são discutidas regras de combinação das saídas dos classificadores do *ensemble*. São discutidas regras não treináveis, como regra do produto e regra da soma; e regras treináveis, onde as saídas dos classificadores base são usadas como entrada para outro algoritmo de aprendizagem que faz a combinação dos resultados dos classificadores base.

[Santos, Sabourin e Maupin 2008] propõem uma estratégia de geração e escolha de classificadores, que combina otimização e seleção dinâmica para permitir a seleção do subconjunto de classificadores mais confiável para rotular cada documento de consulta (teste). O nível de otimização tem como objetivo gerar uma população de conjuntos de classificadores candidatos altamente precisos, enquanto o nível de seleção dinâmico aplica medidas de confiança para revelar o conjunto de candidatos com o mais alto grau de confiança na decisão atual. A novidade do método proposto é dividir a fase de seleção em dois níveis: otimização e seleção dinâmica, conduzindo o segundo nível por meio de medidas de confiança baseadas na extensão de consenso dos conjuntos de classificadores. Dois métodos clássicos de criação de conjuntos, o subespaço aleatório e *Bagging*, foram empregados na fase de geração dos classificadores. As taxas de erro do conjunto e as medidas de diversidade orientaram a otimização. Por fim, três medidas de confiança foram aplicadas no nível de seleção dinâmica: (1) ambiguidade; (2) margem; e (3) força em relação à classe mais próxima. Além disso, a seleção dinâmica do classificador baseado na acurácia local (DCS-LA) foi comparada às estratégias baseadas na confiança.

3.2 Seleção de método de classificação

Para realizar a tarefa de classificação dos documentos adotamos uma estratégia de seleção customizada do classificador. Na seleção customizada é feita a escolha de um classificador ou de um conjunto (*ensemble*) de classificadores para cada documento de consulta. Após a seleção, o classificador ou o conjunto de classificadores selecionado será utilizado para fazer a classificação do documento de consulta. Segundo os autores [Polikar 2006] e [Woods, Kegelmeyer e Bowyer 1997], para a tarefa de ADC um conjunto de classificadores apresenta melhor acurácia quando comparado com um único classificador. As técnicas de DS funcionam estimando o nível de competência de cada classificador a partir de um conjunto de classificadores. Para [Cruz, Sabourin e Cavalcanti 2018] apenas o classificador mais competente, ou um conjunto contendo os classificadores mais competentes, é selecionado para fazer a predição do rótulo de um documento de consulta

específico. De acordo com [Cruz, Sabourin e Cavalcanti 2018], a justificativa para a eficácia das técnicas de seleção dinâmica (DS), em relação ao classificador único, é que nem todos os classificadores do *ensemble* são especialistas em classificar todas os documentos desconhecidos. Cada classificador base é um especialista em uma região local diferente do espaço de características [Zhu, Wu e Yang 2004].

Considerando os trabalhos [Cruz, Cavalcanti e Ren 2011], [Polikar 2006], [Jr, Sabourin e Oliveira 2014] e [Cruz, Sabourin e Cavalcanti 2018] e baseados na realização de experimentos iniciais, onde alcançamos melhores resultados com essa forma de seleção, nossa abordagem utiliza a seleção dinâmica de um conjunto de classificadores. A seguir nosso método é detalhado.

Considere um conjunto de classificadores C e um conjunto de dados D (dividido em treino, validação e teste). Para cada classificador C_i é feito o treinamento utilizando apenas a base de treino. Após a etapa de treinamento é realizada a etapa de validação com predição dos documentos do conjunto de dados de validação. Para cada documento de validação é definido um vetor $V(\mathbf{X}) = \{C_1(\mathbf{X}), C_2(\mathbf{X}), ..., C_L(\mathbf{X})$, onde cada elemento indica se o classificador C_i acertou ou não a predição do documento.

$$C_i(\mathbf{X}) = \begin{cases} 1 & \text{se } C_i(\mathbf{X}) = R(\mathbf{X}) \\ 0 & \text{se } C_i(\mathbf{X}) \neq R(\mathbf{X}) \end{cases}$$

Após finalizar a etapa de validação é feito um novo treinamento, utilizando os conjuntos de dados de treino e validação como dados de treinamento, para cada um dos classificadores base do conjunto de classificadores (*ensemble*). Esse novo treinamento é realizado com o intuito de aumentar a precisão dos classificadores base. Em outras abordagens de seleção dinâmica, descritas na literatura, o treinamento é feito, para cada documento de consulta a ser classificado, com os k vizinhos do documento de consulta. Nossa abordagem utiliza um conjunto de dados mais robusto (dados de treino e validação) e os classificadores base são treinados uma vez (antes de iniciar a etapa de consulta), para classificar todos os documentos de consulta. Dessa forma nossa abordagem utiliza um conjunto de dados maior para o treinamento, além de evitar o treinamento dos classificadores base para cada documento de consulta enviado para o classificador. Com os classificadores treinados é iniciada a etapa de consulta. Para cada documento de consulta são identificados os k vizinhos mais próximos, no conjunto de validação. Após identificar os k vizinhos mais próximos do documento de consulta buscamos, nos vetores gerados na etapa de validação, as informações de acerto ou erro da predição desses documentos vizinhos para cada um dos classificadores base. Os classificadores, que acertaram a predição de pelo menos um dos k vizinhos, são selecionados e utilizados para fazer a predição do documento de consulta. Com as predições realizadas (pelos classificadores selecionados) para o documento de consulta, nosso método, pode realizar uma dos três

formas de *ensemble* a seguir:

- **Votação majoritária** (SMART vm) - escolha da classe que obteve a maior votação entre os classificadores base. Utiliza estratégia *não treinável* de combinação de classificadores, pois avalia as saídas dos classificadores diretamente;
- **Ponderação por similaridade** (SMART ps) - baseado na distância do cosseno e no algoritmo k-NN. Para cada classificador base são identificados os k vizinhos do documento de consulta. Então, apenas para os vizinhos que o classificador acertou a predição, é feita a soma das distâncias (cosseno) entre o documento de consulta e esses vizinhos. O valor final (soma das distâncias) de cada classificador é normalizado pelo maior valor acumulado por um dos classificadores base. Utiliza a estratégia *ponderação dinâmica* para combinação dos classificadores;
- **Ponderação por acerto** (SMART pa) - baseado no algoritmo k-NN. Para cada classificador base são identificados os k vizinhos do documento de consulta. É calculado o número de predições corretas, nos k vizinhos do documento de consulta, para cada classificador base. O valor acumulado, por classificador base, é normalizado pelo maior valor acumulado por um dos classificadores. Utiliza a estratégia *ponderação dinâmica* para combinação dos classificadores base.

3.2.1 Etapas do método proposto

3.2.1.1 Conjuntos de dados

Para avaliar os classificadores, do método proposto neste trabalho, nossos experimentos foram realizados com 6 (seis) conjuntos de dados textuais amplamente utilizados em aplicações do mundo real. São cobertos dois domínios de classificação de documentos, categorização de tópicos e análise de sentimentos. Os conjuntos de dados usados são compostos por documentos de artigos científicos e de notícias. Foi realizada a tarefa de préprocessamnto dos conjuntos de dados, com operações como tokenização, remoção de *stopwords* e cálculo do *tf-idf* ponderado para preparar as coleções para serem usadas na tarefa de classificação. Cada conjunto de dados foi dividido em 10 (dez) partições (*folds*), cada partição (*fold*) separada em 2 (dois) arquivos, sendo um arquivo com dados de treino e outro com dados de teste. No momento do treinamento/validação cada arquivo de treino é dividido em treino e validação, sendo 64% dos dados destinados para treinamento e 36% destinados para validação. A seguir, é feita uma breve descrição a respeito dos conjuntos de dados utilizados:

- **20 News Group (20 ng)** - conjunto de documentos de grupos de notícias, dividido de forma equilibrada em 20 (vinte) classes. 20ng é um conjunto de dados muito popular para experimentos de classificação de documentos;

- **ACM-DL (ACM)** - é um subconjunto da Biblioteca Digital ACM, contendo artigos de Ciência da Computação. Foram consideradas apenas as 11 (onze) classes que correspondem ao primeiro nível da taxonomia adotada pelo ACM;
- **Reuters (REUT90)** - composto de artigos de notícias coletados e classificados pelo Carnegie Group e pela agência Reuters. É uma coleção de documentos desbalanceada, categorizada em 90 (noventa) classes;
- **4 Universities (4UNI), a.k.a. WebKB** - contém páginas da Web coletadas de departamentos de Ciência da Computação de quatro universidades. As páginas foram manualmente classificadas em 7 (sete) classes;
- **Yelp Reviews** - conjunto de dados de avaliações (revisões) *on line*, feitas por usuários, sobre produtos e/ou serviços.

Os detalhes dos conjuntos de dados utilizados são apresentados na tabela 3.1. Nessa tabela são mostradas informações gerais como o número de documentos nos conjuntos de dados e o número de características (termos) dos documentos.

Tabela 3.1: Informações gerais dos conjuntos de dados.

Conjuntos de Dados	# Documentos	# Termos	# Classes	Balanceado
20NG	18.846	49.025	20	Balanceado
REUT90	13.327	17.029	90	Altamente Desbalanceado
4UNI	8.199	22.581	7	Desbalanceado
ACM	24.897	23.110	11	Desbalanceado
YELP_REVIEWS	5.000	21.940	2	Balanceado
STANFORD	359	1333	2	Balanceado

3.2.1.2 Conjunto de classificadores

A estratégia usada em sistemas de múltiplos classificadores é definir vários classificadores e combinar suas saídas para fazer a predição final do documento de consulta. A escolha do conjunto inicial de classificadores é fundamental para que esta abordagem consiga alcançar melhores resultados que um classificador individual. Para definir os classificadores base, do conjunto de classificadores, foi realizada uma análise entre os métodos *SGD*, *Logistic Regression*, *SVM*, *Naive Bayes*, *k-NN* e *Random Forest*. A abordagem usada para escolher os classificadores que compõem o conjunto inicial de classificadores foi baseada na precisão e diversidade dos classificadores base.

Segundo [Kuncheva 2014] a intuição comum sugere que os classificadores em um *ensemble* devem ser tão precisos quanto possível e não devem cometer erros coincidentes (não devem errar a predição dos rótulos para os mesmos documentos classificados). De acordo com [Polikar 2006], a estratégia em MCS é juntar vários classificadores e combinar suas saídas, de uma forma que a combinação melhore o desempenho de um único classificador. Isso requer, no entanto, que classificadores individuais errem as predições para diferentes documentos. Ainda de acordo com [Polikar 2006], se cada classificador

erra a predição para diferentes documentos, uma combinação estratégica desses classificadores pode reduzir o erro total. A ideia é, portanto, tornar cada classificador o mais único possível, particularmente com respeito a documentos classificados incorretamente.

Dessa forma foi possível escolher classificadores que alcançam um alto nível de precisão e que são o mais específico possível em relação aos documentos classificados incorretamente. Mais precisamente, precisamos de classificadores cujos limites (fronteiras) de decisão, de um classificador, sejam diferentes dos limites de decisão de outro classificador. Esse conjunto de classificadores é considerado diverso [Polikar 2006]. Para [Kuncheva 2014], os limites (fronteiras) de decisão estão localizados nas regiões de transição (separação) entre os documentos de classes diferentes. Divide o espaço vetorial em conjuntos, um para cada classe de documentos.

O cálculo da precisão e da diversidade dos classificadores base foi realizada de maneira empírica fazendo o treinamento dos classificadores individualmente em diferentes conjuntos de dados de treino. Com os resultados obtidos nos treinamentos foi realizada análise para verificar quais os classificadores alcançaram maior precisão e melhor nível de diversidade entre os classificadores. Os classificadores base, com melhores resultados em precisão e diversidade, foram escolhidos para compor o *ensemble*, que é usado pelo nosso método, para a tarefa de ADC. O cálculo da diversidade foi baseado na medida em pares [Kuncheva 2014], que compara os limites de decisão de cada par de classsificadores por vez, para encontrar a diversidade entre estes classificadores.

A seleção dos classificadores base, de acordo com o critério de acurácia, foi feita utilizando os resultados da tabela 3.2, seguindo as seguintes etapas:

1. Treinamento dos classificadores base em 6 (seis) conjuntos de dados, descritos tabela na 3.1;
2. Avaliação da precisão de cada classificador base no conjunto de dados de validação;
3. Escolha dos classificadores base mais precisos.

Tabela 3.2: Precisão dos classificadores base nos conjuntos de dados de validação.

Conjuntos de Dados	# SGD	# Logistic	# SVM	Naive Bayes	kNN	Random Forest
20NG	0.88	0.87	0.87	0.88	0.84	0.85
REUT90	0.69	0.70	0.65	0.64	0.65	0.62
4UNI	0.80	0.80	0.81	0.68	0.77	0.74
ACM	0.77	0.77	0.76	0.73	0.73	0.71
YELP_REVIEWS	0.94	0.94	0.94	0.90	0.89	0.83
STANFORD	0.78	0.78	0.79	0.79	0.75	0.79

As medidas de diversidade são calculadas usando uma tabela de contingência que resume o comportamento de dois classificadores C_i e C_j em um conjunto de dados. A tabela 3.3 apresenta um exemplo de tabela de contingência. Os valores na tabela significam: N^{11} é a fração de documentos classificados corretamente por C_i e C_j; N^{10} é a

fração de documentos classificados corretamente por C_i, mas incorretamente classificadas por C_j; N^{01} é a fração de documentos incorretamente classificados por C_i e corretamente classificados por C_j; N^{00} é a fração de documentos incorretamente classificados por C_i e C_j. Considere, $N^{11} + N^{10} + N^{01} + N^{00} = 1$. Para obter a diversidade geral para o conjunto de classificadores, foi calculada a média das medidas dos pares.

Tabela 3.3: Tabela de contingência para os classificadores C_i e C_j.

	$C_j = 1$	$C_j = 0$
$C_i = 1$	N^{11}	N^{10}
$C_i = 0$	N^{01}	N^{00}

A seleção dos classificadores base, de acordo com o critério de diversidade, foi feita utilizando os resultados (acertos e erros nas predições) obtidos no conjunto de validação. Os cálculos foram realizados de acordo com a tabela de contingência 3.3, utilizando a equação 3-1, e seguiram as seguintes etapas:

1. Treinamento dos classificadores base em 6 (seis) conjuntos de dados;
2. Predição dos documentos, do conjunto de dados de validação, para todos os classificadores base;
3. Cálculo da diversidade entre pares de classificadores base, utilizando as predições feitas na etapa 2;
4. Cálculo da diversidade média entre os classificadores base;
5. Escolha dos classificadores base com maior diversidade.

Para calcular a diversidade entre um par de classificadores é utilizada a fórmula 3-1. Onde Q assume valores positivos se as mesmas instâncias de documentos forem classificadas corretamente pelos dois classificadores do par; e valores negativos, caso contrário. A diversidade máxima é obtida para $Q = 0$.

$$Q_{i,k} = \frac{N^{11}.N^{00} - N^{01}.N^{10}}{N^{11}.N^{00} + N^{01}.N^{10}} \tag{3-1}$$

Após realizar a análise dos resultados foram escolhidos os classificadores que obtiveram maior precisão e maior diversidade (que apresentaram erro de predição em diferentes documentos). Com a avaliação, foram escolhidos os seguintes classificadores base:

1. *SGD*;
2. *Logistic Regression*;
3. *SVM*;
4. *Naive Bayes Multinomial*;
5. *Random Forest*;
6. *k-NN*.

3.2.1.3 Grade de pesquisa (GridSearch)

Com o intuito de ajustar o desempenho dos classificadores (modelos de aprendizado), escolhidos para compor o *ensemble* proposto neste trabalho, foi utilizada a técnica de pesquisa em grade *GridSearch* para selecionar a melhor configuração de hiperparâmetros para cada um dos classificadores base utilizado. Esta técnica recebe um classificador e uma grade de hiperparâmetros (ex. 10 parâmetros com 5 valores diferentes para cada parâmetro), então são executadas várias rodadas de treinamento para conseguir ajustar os melhores hiperparâmetros para o modelo.

Foi utilizado o método *GridSearchCV*, da biblioteca *Scikit Learn*, que faz uma pesquisa exaustiva sobre todos os valores de hiperparâmetros (grade de hiperparâmetros) especificados para um classificador. O método *GridSearchCV* implementa uma técnica de ajuste dos hiperparâmetros e uma técnica de pontuação para conseguir selecionar os melhores hiperparâmetros. Após testar todas as possibilidades de valores de hiperparâmetros para um classificador, em um determinado conjunto de dados de treino, são retornados os valores que obtiveram o melhor desempenho de acordo com uma medida especificada, como por exemplo, precisão. Em nossos experimentos o *GridSearch* foi executado para cada um dos classificadores em todos os conjuntos de dados utilizados, com validação cruzada de 3 partições (*folds*).

A seguir estão relacionados hiperparâmetros utilizados para executar o *GridSearch* para cada um dos classificadores base.

- *SGD* - variamos a função de perda *loss* $\in$ {*hinge*, *modified_huber*, *squared_hinge*}, a penalidade *penalty* $\in$ {*l1*, *l2*} e a taxa de aprendizagem *learning_rate* $\in$ {*constant*, *optimal*, *adaptive*, *invscaling*};
- *Logistic Regression* - variamos tanto a norma usada na penalização *penalty* $\in$ {*l1*, *l2*} quanto a parâmetro de penalidade $C \in \{0.001, 0.01, 0.1, 1, 10\}$;
- *SVM* - variamos o *kernel* $\in$ {*rfb*, *linear*} e a penalidade $C \in \{0.0001, 1, 10, 25\}$;
- *Naive Bayes Multinomial* - variamos o parâmetro de suavização *alpha* $\in$ {1, 0.1, 0.01, 0.001, 0.0001, 0.00001};
- *Random Forest* - variamos o número de árvores {100, 200 e 300}, o critério de divisão *criterion* $\in$ {*'gini'*, *'entropy'*} e o número de recursos a serem considerados ao procurar a melhor divisão *max_features* $\in$ {*'sqrt'*, *0.3*, *'log2'*}. Para melhorar a acurácia do algoritmo *Random Forest*, foi avaliada a utilização de um número maior de árvores nos experimentos com este algoritmo. Mas por questão de tempo estes experimentos não foram realizados;
- *k-NN* - experimentamos diferentes tamanhos de vizinhança *n_neighbors* $\in$ {5, 10, 30, 100, 200}, métricas de distância *metric* $\in$ {*'cosine'*, *'euclidean'*} e função de peso usada na previsão *weights* $\in$ {*'uniform'*, *'distance'*}.

3.2.1.4 Etapa de treinamento

O treinamento do modelo envolve alimentar o algoritmo com os dados de treinamento e seus respectivos rótulos, de modo que o algoritmo seja capaz de aprender vários padrões correspondentes a cada classe e possa reutilizar esse conhecimento para prever classes de novos documentos. Freqüentemente, um conjunto de dados de validação, opcional, é usado para avaliar o desempenho do algoritmo de classificação para garantir que ele generalize bem os dados durante o treinamento.

Com os classificadores base escolhidos, os hiperparâmetros selecionados e os conjuntos de dados definidos é feita a construção dos modelos de aprendizado do método proposto neste trabalho. Para construir os modelos foi realizado treinamento para cada classificador do *ensemble* nos conjuntos de dados, figura 3.3. Nesta etapa foram feitos três experimentos diferentes. O primeiro utilizou a abordagem *leave-one-out*, o segundo *leave-p-out* e o terceiro fez a divisão do conjunto de dados de treinamento em treino (64%) e validação (36%). Os três experimentos foram realizados com a utilização de validação cruzada de 10 (dez) folds. Feita a análise dos resultados optou-se pelo terceiro experimento (divisão dos conjuntos de dados de treinamento em treino e validação), pois este experimento teve desempenho computacional muito superior aos demais e os resultados de precisão entre as três abordagens foram muito próximas.

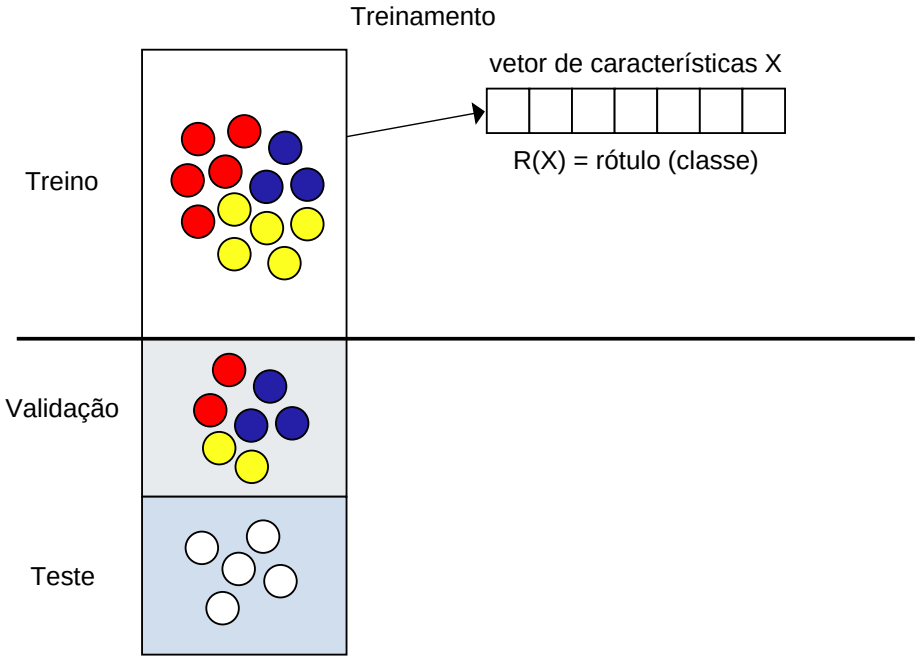
Como citado na seção 3.2.1.1, cada conjunto de dados foi dividido em 10 partições (*folds*) e cada partição dividida em treino (75%) e teste (25%). Nesta etapa a base de treino foi subdividida em treino e validação, esta abordagem é usada para avaliar o desempenho do algoritmo de classificação para garantir que o algoritmo seja capaz de generalizar bem os dados. Então, para um determinado conjunto de dados todos os classificadores são treinados e os respectivos modelos de aprendizado são construídos.

A seguinte sequência de passos é realizada, na etapa de treinamento, para cada uma das 10 (dez) partições dos conjuntos de dados:

1. Leitura dos conjuntos de dados de treinamento;
2. Divisão dos conjuntos de dados de treinamento em treino (64%) e validação (36%);
3. Treinamento dos classificadores utilizando o conjunto de dados de treino (64%).

Os modelos aprendidos são usados na próxima etapa (validação) do método.

Figura 3.3: Etapa de treinamento. Cada classificador recebe os dados de treino e seus respectivos rótulos



3.2.1.5 Etapa de validação

Após o treinamento dos classificadores base é realizada a etapa de validação. Nesta etapa é feita a predição de todos os documentos, do conjunto de dados de validação, utilizando os classificadores que foram treinados na etapa de treinamento. Para cada documento de validação é definido um vetor $V(\mathbf{X}) = \{C_1(\mathbf{X}), C_2(\mathbf{X}), ..., C_L(\mathbf{X})$, onde cada elemento indica se o classificador C_i acertou ou não a predição do documento. A figura 3.4 apresenta uma visão geral desta etapa.

$$C_i(\mathbf{X}) = \begin{cases} 1 & \text{se } C_i(\mathbf{X}) = R(\mathbf{X}) \\ 0 & \text{se } C_i(\mathbf{X}) \neq R(\mathbf{X}) \end{cases}$$

Como resultado dessa etapa é produzida uma matriz, exemplo representado pela tabela 3.4, onde cada linha corresponde aos acertos (1) e erros (0), das predições feitas pelos classificadores base, para um determinado documento do conjunto de dados de validação. Essas informações são usadas na etapa de consulta para seleção customizada dos classificadores.

Figura 3.4: Etapa de validação.

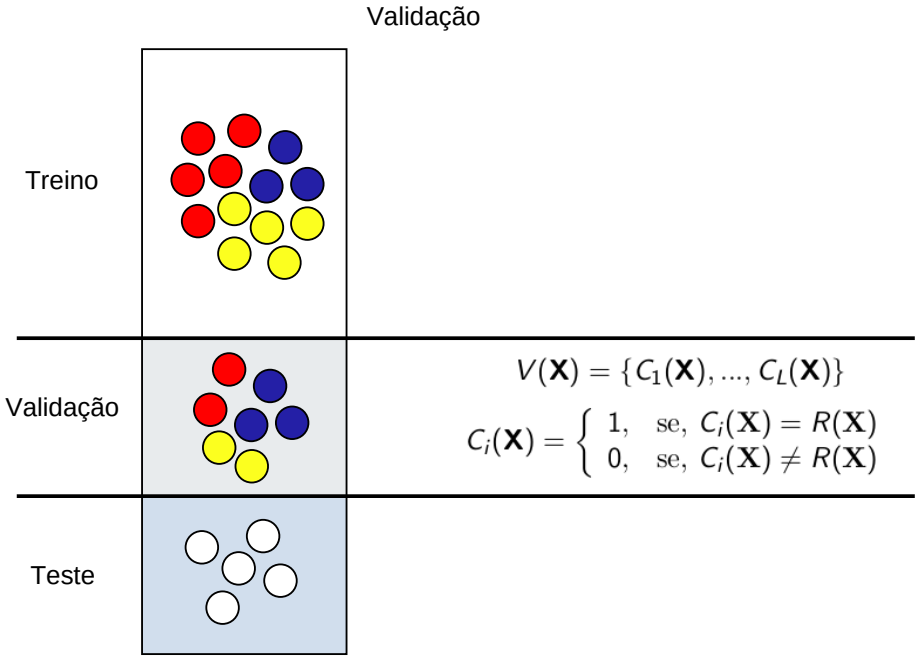


Tabela 3.4: Exemplo de matriz com as predições dos documentos do conjunto de dados de validação.

	$C_1(\mathbf{X})$	$C_2(\mathbf{X})$	$C_3(\mathbf{X})$	$\cdots$	$C_L(\mathbf{X})$
$V_1(\mathbf{X})$	1	0	0	1	1
$V_2(\mathbf{X})$	0	0	0	1	1
$V_3(\mathbf{X})$	1	0	1	0	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$V_n(\mathbf{X})$	1	1	0	1	1

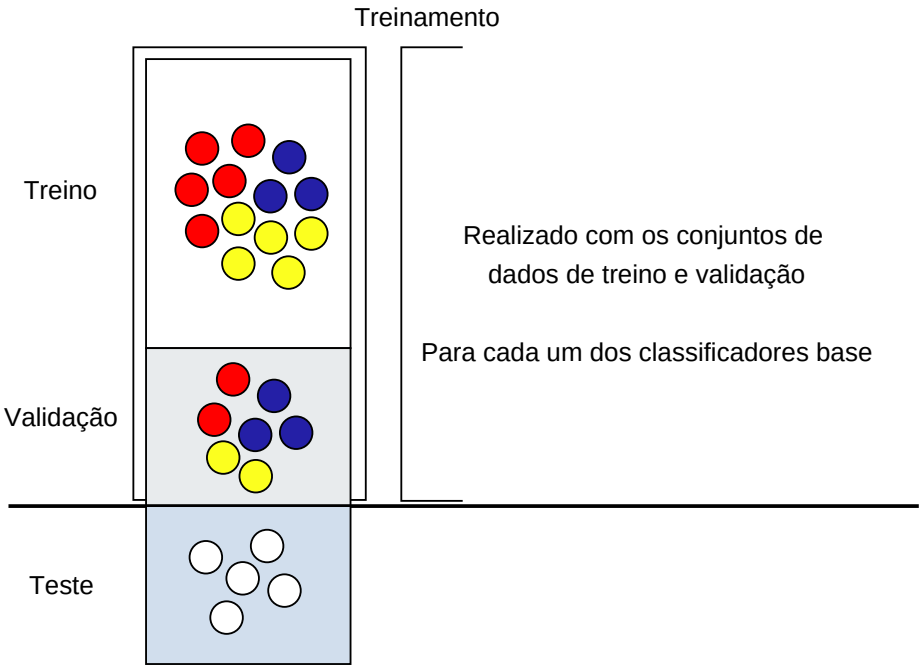
3.2.1.6 Segunda etapa de treinamento

Uma das contribuições do nosso método é a segunda etapa de treinamento, que é realizada após a etapa de validação e antes de iniciar a etapa de consulta. Este segundo treinamento é realizado de forma diferente do primeiro treinamento. Enquanto no primeiro treinamento os dados de treinamento são divididos em treino (64%) e validação (36%), no segundo treinamento não há separação. Ou seja, o conjunto de dados de treinamento é usado inteiramente para treinar os classificadores base. O treinamento continua sendo realizado para cada um dos classificadores base do *ensemble*, com os mesmos hiperparâmetros selecionados na fase de *GridSearch* e com validação cruzada de 10 (dez) *folds*. O propósito de realizar o segundo treinamento é melhorar a precisão dos classificadores base, para a predição dos documentos de consulta, na próxima etapa do nosso método. Quanto mais dados tivermos para treinar o algoritmo mais alto poderá ser

o nível de precisão deste algoritmo. Então, nossa abordagem utiliza todo o conjunto de dados de treino para realizar o treinamento dos classificadores base.

Dessa forma, nosso método deixa os classificadores base pré-treinados para realizar a próxima etapa do método (etapa de predição dos documentos de consulta). Com esta técnica, nosso método faz um único treinamento, dos classificadores base, para todos os documentos de teste que serão classificados. Nosso método diferencia-se das abordagens encontradas na literatura, como por exemplo, o método *MetaLazy* proposto por [Mendes L. F. 2020]. Pois estas abordagens ao receberem cada documento de consulta para ser classificado, identificam os k vizinhos do documento de consulta e então realizam o treinamento dos classificadores base utilizando a vizinhança identificada, como conjunto de dados de treino. Ou seja, o treinamento é feito para cada documento de consulta. Uma visão geral desta etapa do nosso método é apresentada na figura 3.5.

Figura 3.5: Segunda etapa de treinamento.



3.2.1.7 Etapa de consulta

Nossa abordagem inicia a etapa de consulta (teste) com os classificadores base pré-treinados. Esta abordagem proporciona um ganho de desempenho em relação aos métodos *Lazy*, já que estes métodos treinam os classificadores base após receberem um documento de consulta para ser classificado e geralmente com a utilização da vizinhança (k-NN) do documento de consulta como conjunto de treino.

O método proposto faz a seleção customizada do classificador ou subconjunto de classificadores baseada na região local de competência do classificador base. Nossa abordagem utiliza o conjunto de validação para identificar os k vizinhos do documento de consulta (região local) e selecionar o classificador mais competente ou o subconjunto de classificadores mais competentes. A particularidade desse conjunto de validação é que, para cada um dos seus documentos, temos a informação de quais classificadores do *ensemble* acertaram a sua predição. As informações das predições, dos documentos do conjunto de validação, são armazenadas na etapa de validação, conforme exemplo apresentado na tabela 3.4. Baseados nessas informações realizamos a seleção customizada do(s) classificador(es).

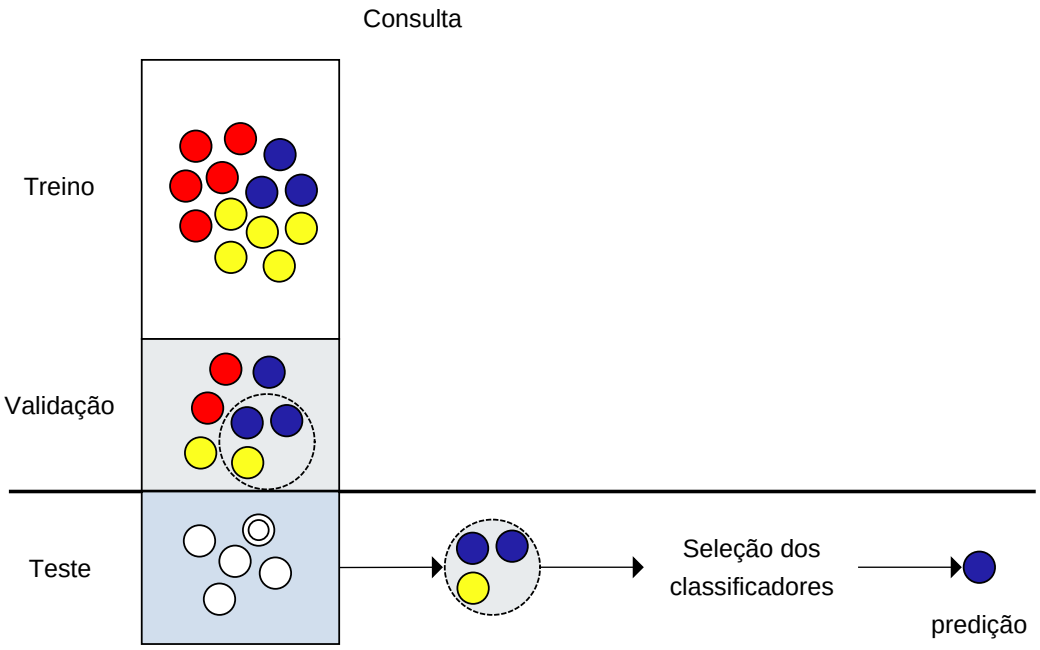
Na seleção dinâmica proposta pelo presente trabalho considere X_i, $i = \{1, ..., k\}$ como sendo os k vizinhos mais próximos do documento de consulta X e considere um conjunto de L classificadores C_j, $j = \{1, ..., L\}$. O subconjunto de classificadores dinâmicos selecionados C^*, para classificar o documento de consulta, é composto pelos classificadores C_j que classificam corretamente pelo menos um dos k vizinhos de X. Classificadores que não classificam corretamento pelo menos um dos k vizinhos de X_i são desconsiderados para compor o *ensemble* que fará a predição do documento de consulta.

Para cada documento de consulta, recebido pelo método, são identificados os k vizinhos mais próximos do documento de consulta, na base de dados de validação. Com os k vizinhos identificados buscamos nos vetores, gerados na etapa validação $V(\mathbf{X}) = \{C_1(\mathbf{X}), C_2(\mathbf{X}), ..., C_L(\mathbf{X})$, quais classificadores acertaram as predições para esses vizinhos. Os classificadores, que acertaram a predição de pelo menos um dos k vizinhos, são selecionados e usados para fazer a predição do documento de consulta. Com as informações das predições nosso método realiza a combinação dos resultados dos classificadores para fazer a predição final do documento de consulta. Na figura 3.6 é apresentada esta etapa do método. Nosso método utiliza a combinação de rótulos de classes, onde apenas os rótulos das classes são avalidadas nas saídas dos classificadores. Com os rótulos das classes nosso método pode realizar uma das três formas de *ensemble* a seguir:

- **Votação majoritária** (SMART vm) - escolha da classe com maior votação;
- **Ponderação por similaridade** (SMART ps) - soma as distâncias entre o documento de consulta e seus vizinhos, para cada classificador. Normalizado pelo maior valor acumulado;
- **Ponderação por acerto** (SMART pa) - calcula o número de predições corretas nos k vizinhos, para cada classificador. Normalizado pelo maior valor acumulado.

A descrição completa das três formas de *ensemble* é apresentada na seção 3.2.

Figura 3.6: Etapa de consulta (teste).



Durante os experimentos foi verificado que, para alguns documentos de consulta, os classificadores base não acertam a predição de nenhum dos seus k vizinhos. Quando isso ocorre, o método proposto seleciona a classe que foi mais votada, entre os k vizinhos, e a atribui ao documento de consulta.

3.2.2 Paralelismo

A classificação automática de documentos é uma tarefa que requer um alto custo computacional pois, no caso da abordagem *Lazy*, deve operar no momento da classificação e além disso envolve o uso de grandes conjuntos de dados. Como forma de acelerar o método proposto, foi utilizado o paralelismo em determinadas partes da implementação com o uso do *Scikit-Learn* (biblioteca de aprendizado de máquina de código aberto para linguagem *Python*). Essa biblioteca possibilita o uso de paralelismo com alteração de parâmetros no momento da chamada de seus métodos.

Outra abordagem testada para acelerar o desempenho do nosso método foi a utilização da biblioteca *RAPIDS cuML*, que implementa o paralelismo de granularidade fina em aceleradores gráficos (*GPU*). Esta abordagem está em andamento e tem grande potencial para acelerar as operações do nosso método.

3.3 Experimentos realizados e discussões

Para avaliar o método SMART consideramos conjuntos de dados variados (tamanho, assimetria, esparsidade e número de classes), de duas tarefas importantes da classificação de documentos, categorização de tópicos e análise de sentimentos. As informações gerais a respeito dos conjuntos de dados são apresentadas na tabela 3.1.

Abaixo são apresentados alguns parâmetros utilizados nos experimentos.

- Foram utilizados os seguintes classificadores para construir o *ensemble*: *SGD*, *Logistic Regression*, *SVM*, *Naive Bayes Multinomial*, *Random Forest* e *k-NN*;
- Método *SMART vm*: *ensemble* com votação majoritária;
- Método *SMART ps*: *ensemble* com ponderação por similaridade;
- Método *SMART pa*: *ensemble* com ponderação por acerto;
- Foi realizado *GridSearch* para encontrar os melhores hiperparâmetros para um dado classificador em cada conjunto de dados;
- Os conjuntos de dados foram separados em treino, validação e teste;
- Foi utilizada validação cruzada com 10 *folds*;
- Foram usados valores de k = {5, 10, 100, 200} para encontrar os vizinhos do documento de consulta;

Na avaliação do nosso método (SMART) com a *baseline* (*MetaLazy*) foram utilizadas duas métricas padronizadas, amplamente usadas para avaliar classificação de documentos, *macro averaged F1* (*macroF1*) e *micro averaged F1* (*microF1*). As duas métricas são baseadas nas métricas *Precisão* (*Precision*) e *Revocação* (*Recall*). A *macroF1* mede a eficácia da classificação, calculando a média aritmética simples da métrica *F1*pdsa entre todas as classes do conjunto de dados. Enquanto a *microF1* calcula a fração dos documentos classificados corretamente em cada classe.

Os resultados obtidos são apresentados na tabela 3.5. Comparamos as versões do método proposto SMART com o método *MetaLazy* (*baseline*) e com outros métodos descritos na literatura, em especial incluímos o método BERT [Devlin et al. 2018] (do Google) que vem recebendo bastante atenção recentemente. Os melhores resultados são apresentados em negrito. Como pode ser observado nos resultados, nenhum dos classificadores consegue ter o melhor desempenho em todas as situações experimentadas.

Tabela 3.5: Resultados dos experimentos.

		20NG	REUT90	4UNI	ACM	YELP	STANFORD	Posicionamento Médio
SMART *vm*	macroF1	89.68 (4)	38.79 (3)	73.58 (3)	68.37 (3)	94.94 (4)	81.21 (5)	3.6
	microF1	89.97	**70.06**	**82.97**	**79.33**	94.94	81.33	
SMART *ps*	macroF1	89.77 (2)	**39.72** (1)	**74.54** (1)	**69.16** (1)	95.00 (2)	81.25 (3)	1.8
	microF1	90.06	**70.19**	**83.02**	**79.28**	95.00	81.33	
SMART *pa*	macroF1	89.75 (3)	**39.41** (2)	**74.52** (2)	**69.14** (2)	95.00 (3)	81.25 (4)	2.6
	microF1	90.03	**70.02**	**83.07**	**79.10**	95.00	81.33	
MetaLazy	macroF1	**91.17** (1)	33.89 (5)	60.89 (6)	65.07 (5)	91.39 (6)	**83.18** (1)	4.0
	microF1	**91.46**	66.96	77.99	78.74	91.40	**83.27**	
BERT	macroF1	84.05 (6)	35.37 (4)	69.58 (5)	59.25 (6)	**97.26** (1)	**83.07** (2)	4.0
	microF1	84.39	67.57	**83.64**	76.46	**97.26**	**83.31**	
SVM	macroF1	88.86 (5)	33.55 (6)	71.57 (4)	67.13 (4)	94.32 (5)	81.20 (6)	5.0
	microF1	89.01	68.51	80.71	76.80	94.32	81.30	

Considerando os conjuntos de dados desbalanceados (REUT90, 4UNI e ACM), observamos que tanto o método *MetaLazy* quanto o método SMART foram capazes de identificar corretamente documentos pertencentes às classes menores (com menos documentos), sem deixar de fazer as predições corretas para as classes maiores (classes com mais documentos). No trabalho *MetaLazy*, proposto por [Mendes L. F. 2020], é descrito que o seu método identifica corretamente documentos pertencentes tanto às classes com menos documentos quanto às classes com mais documentos. Para compararmos os dois métodos, e sabermos como o método SMART se comporta em relação à predição dos documentos por classe, foi feita uma contagem de acertos e erros das predições dos documentos para cada classe, em cada um dos conjuntos de dados utilizados nos experimentos.

Esta avaliação indica que os dois métodos possuem alto desempenho em termos da métrica *macroF1*, além de produzirem resultados competitivos na métrica *microF1*. A tabela 3.5, coluna Posicionamento Médio, apresenta o posicionamento de cada classificador, baseado na métrica *macroF1*, em cada um dos conjuntos de dados. Esses resultados demostram a consistência do método SMART (principalmente o SMART *ps*) que apresenta resultados competitivos em 5 conjuntos de dados dos 6 avaliados. O valor da coluna Posicionamento Médio é calculado da seguinte forma: soma-se os valores de cada linha (posicionamento do classificador no conjunto de dados) e divide pelo número de conjuntos de dados utilizados. O classificador que obteve o menor valor de média é o que apresenta melhor posicionamento médio. Ou seja, é o classificador que obteve o melhor resultado de *macroF1* na média, para todos os conjuntos de dados utilizados.

3.3.1 Tempo de execução

Nesta seção, analisamos o método SMART em termos de tempo de execução. Primeiramente, comparamos o tempo de execução (GridSearch + Treinamento) do método SMART com o tempo do *MetaLazy* (*baseline*) e dos demais métodos dos experimentos. Depois fazemos uma comparação de tempo de consulta (Predição) entre os mé-

todos. A primeira comparação da análise (GridSearch + Treinamento) consiste no tempo médio gasto para encontrar os melhores hiperparâmetros para os classificadores base e treinar cada classificador com o conjunto de dados de treinamento. A segunda comparação (Predição), consiste no tempo médio para fazer a classificação de um documento de consulta. Os métodos *MetaLazy* e SMART foram implementados utilizando a técnica de validação cruzada. Portanto, para medir o tempo de execução nos dois casos foram feitos os seguintes cálculos:

- **GridSearch + Treinamento** - para cada conjunto de dados somamos o tempo para executar cada *fold* e, em seguida, dividimos pelo número total de *folds*;
- **Predição** - para cada conjunto de dados somamos o tempo para executar cada *fold* e, em seguida, dividimos pelo número total de *folds* e depois dividimos pelo número de documentos do conjunto de teste.

Os experimentos dos métodos SMART e *MetaLazy* foram realizados utilizando uma máquina com Intel Xeon CPU E5-2620 com 12 núcleos, 32GB de memória RAM e quatro placas gráficas (GPU) Ge-Force GTX Titan Black com 6GB de memória. A linguagem de programação usada foi Python (3.5) com biblioteca *scikit-learn* (0.20.3). Os métodos BERT e SVM foram executados em uma máquina máquina muito similar, com CPU Intel(R) Core(TM) i7-5820K 3.30GHz com 64GB de memória RAM e uma placa gráfica (GPU) K40c com 12GB de memória.

A tabela 3.6 apresenta os resultados de tempo de execução do GridSearch + Treinamento. Nesta comparação apresentamos os resultados apenas do SMART *ps*, porque os valores de tempo de execução de GridSearch + Treinamento são os mesmos para as três formas de *ensemble* do nosso método. É preciso observar que o tempo de GridSearch + Treinamento é computado em uma etapa de pré-processamento. Outra observação, que pode ser feita, é que por ser utilizada a grade de pesquisa (*GridSearch*), quanto mais parâmetros forem informados para a grade mais tempo será gasto nesta etapa. Isto influencia no tempo de execução de cada um dos métodos comparados.

Tabela 3.6: Tempo médio de execução do GridSearch + Treinamento (segundos).

	20NG	REUT90	4UNI	ACM	YELP	STANFORD
SMART *ps*	13543	3695	2360	4580	1762	164
MetaLazy	8326	11804	1526	5175	522	141
BERT	79678	25023	24110	32687	9467	702
SVM	5549	4058	1196	3135	172	1

Inicialmente, para construir o *ensemble* do método proposto, foram utilizados seis classificadores (*SGD*, *Logistic Regression*, *SVM*, *Naive Bayes Multinomial*, *Random Forest* e *k-NN*), mas com os experimentos foi observado que os métodos *Random Forest* e *k-NN* tem um alto custo computacional. Por isso, foram implementadas duas versões

do método SMART, uma com os 6 (seis) classificadores base, citados acima, e outra com 4 (quatro) classificadores base. Na versão com 4 (quatro) classificadores base foram retirados os algoritmos *Random Forest* e *k-NN*. Um detalhamento, sobre as versões do método proposto, com 4 (quatro) e com 6 (seis) classificadores, é apresentado na seção 3.3.3.

A versão final, da implementação do método SMART, que é utlizada para comparar com as *baselines*, ficou com quatro classificadores base. Essa abordagem proporcionou um ganho de desempenho de execução de até *5x*, em relação ao método *MetaLazy*. O ganho de execução foi conseguido no tempo de consulta (Predição), sendo que o ganho de *5x* foi alcançado nos experimentos com o conjunto de dados STANFORD. Os resultados alcançados são apresentados na tabela 3.7.

Tabela 3.7: Tempo médio de execução da Predição (segundos).

	20NG	REUT90	4UNI	ACM	YELP	STANFORD
SMART *ps*	0.041	0.0353	0.0211	0.0205	0.0102	0.0050
MetaLazy	0.0936	0.1140	0.0561	0.0731	0.0312	0.0260
BERT	0.0001	0.0022	0.0044	0.0301	0.0009	0.0002
SVM	0.0051	0.0040	0.0030	0.0022	0.0006	0.0001

É importante considerar que o *MetaLazy* é um aprendiz postergado (*Lazy Learner*) enquanto o método SMART é considerado *semi-lazy*. Os dois métodos tem abordagens diferentes para a etapa de consulta (predição), onde para cada documento de consulta, são executados os seguintes passos:

- *MetaLazy*
    1. Identfica os *k* vizinhos do documento de consulta e suas distâncias;
    2. Calcula os pesos baseados nas distâncias;
    3. Adiciona novas coocorrências de características nos dados de treinamento e no documento de consulta;
    4. Treina o classificador usando, como conjunto de treinamento, os *k* vizinhos do documento de consulta e classifica o documento de consulta. Utiliza poucos documentos (*k*) para treinar o classificador.

- *SMART*
    1. Idenfica os *k* vizinhos do documento de consulta e suas distâncias;
    2. Seleciona o(s) classificador(es) que acertaram a predição de pelo menos um dos *k* vizinhos;
    3. Faz a predição do documento de consulta com o(s) classificador(es) selecionado(s). Identifica a classe mais votada entre os classificadores selecionados (SMART *vm*); ou calcula os pesos para cada classificador selecionado (SMART *ps* e SMART *pa*);

4. Faz a classificação do documento de consulta, combinando as predições dos classificadores selecionados.

Na etapa de consulta (Predição), pode ser observada uma diferença, entre os métodos *MetaLazy* e SMART, que impacta no tempo de execução, fazendo com o que o método SMART obtenha ganho de desempenho em relação ao *MetaLazy*. Enquanto o *MetaLazy* faz o treinamento dos classificadores base, com os k vizinhos mais próximos do documento de consulta, para cada documento de consulta. O método SMART faz um pré-treinamento dos classificadores base antes de receber os documentos de consulta. Os classificadores base são pré-treinados em uma etapa anterior à etapa de consulta, conforme descrito na seção 3.2.1.6

3.3.2 Testes estatísticos

Foi realizada uma comparação estatística, entre os métodos *MetaLazy* e SMART, com a utilização do *t*-teste pareado com validação cruzada em *k-folds*. A hipótese nula a ser testada é que para um determinado conjunto de dados e dois métodos de classificação, não há significância entre os resultados de precisão (acurácia) destes métodos, ou seja, comprovar que os métodos não possuem diferença significativa entre os seus resultados.

Para executar os testes foi utilizado um programa desenvolvido na linguagem *Python*. Foram informados os seguintes parâmetros para o programa:

- Tabela com os resultados de *macroF1* dos métodos comparados;
- Tabela com os resultados de *microF1* dos métodos comparados;
- Intervalo de confiança de 95% e nível de significância de 0.05 (100% - 95% = 5%);

A execução dos testes estatísticos comparou a versão SMART *ps* com 4 (quatro) classificadores base e o método *MetaLazy*. Foi utilizada a versão SMART *ps* porque é a versão, do nosso método, que apresenta os melhores resultados de precisão. Para fazer o cálculo dos testes estatísticos, foi executado o *t*-teste pareado com validação cruzada em 10 *folds*, para cada um dos conjuntos de dados dos experimentos. A tabela 3.8 apresenta as seguintes informações para os métodos SMART *ps* e *MetaLazy* em cada conjunto de dados dos experimentos:

- Resultados de precisão para as métricas *macroF1* e *microF1*;
- Linha $p_A - p_B$ - diferenças entre as precisões dos métodos SMART *ps* e *MetaLazy* em cada *fold* do conjunto de dados separado por métrica. Onde, p_A é o valor da precisão para o método SMART *ps* e p_B é o valor da precisão para o método *MetaLazy*;

- Coluna *t*-valor - valor calculado com a equação 2-19 para um determinado conjunto de dados e métrica de avaliação (*macroF1* ou *microF1*). Este valor é usado para indicar se os algoritmos testados possuem diferença estatística ou não.

Tabela 3.8: Resultados do *t*-teste pareado calculado com *macroF1* (%) e *microF1*(%).

20NG												
	Métrica	1	2	3	4	5	6	7	8	9	10	t-valor
SMART *ps*	*macroF1*	91.40	88.70	90.03	89.70	89.90	90.40	89.40	89.30	89.40	89.00	
MetaLazy		92.05	90.23	90.74	91.13	90.96	91.84	90.89	91.10	91.89	90.90	
p_A - p_B		-0.65	-1.53	-0.71	-1.43	-1.06	-1.44	-1.49	-1.8	-2.49	-1.90	8.2878
SMART *ps*	*microF1*	91.50	89.10	90.60	90.00	90.01	90.50	89.90	89.50	89.80	89.10	
MetaLazy		92.23	90.63	91.05	91.41	91.25	92.08	91.28	91.27	92.38	91.04	
p_A - p_B		-0.73	-1.53	-0.45	-1.41	-1.24	-1.58	-1.38	-1.77	-1.58	-1.94	9.4356
REUT90												
	Métrica	1	2	3	4	5	6	7	8	9	10	t-valor
SMART *ps*	*macroF1*	40.30	35.90	40.20	40.30	41.00	36.00	39.90	36.70	41.50	40.20	
MetaLazy		36.89	27.76	33.05	32.29	36.31	32.06	38.25	31.67	35.30	35.34	
p_A - p_B		3.41	8.14	7.15	8.01	4.69	3.94	1.65	5.03	6.20	4.86	8.0609
SMART *ps*	*microF1*	69.60	69.20	68.70	71.20	71.00	69.10	72.10	70.90	69.50	71.30	
MetaLazy		66.66	67.46	65.40	67.32	67.39	65.54	68.54	67.90	65.95	67.44	
p_A - p_B		2.94	1.74	3.30	3.88	3.61	3.56	3.56	3.00	3.55	3.86	16.5068
4UNI												
	Métrica	1	2	3	4	5	6	7	8	9	10	t-valor
SMART *ps*	*macroF1*	78.10	72.60	69.80	72.40	73.50	72.00	72.50	76.70	78.30	76.40	
MetaLazy		60.92	58.86	60.40	59.59	61.19	57.41	58.34	64.14	63.49	64.56	
p_A - p_B		17.18	13.74	9.40	12.81	12.31	14.59	14.16	12.56	14.81	11.84	20.2583
SMART *ps*	*microF1*	84.10	81.90	81.20	82.30	83.70	83.50	84.20	82.60	85.20	81.80	
MetaLazy		78.00	77.12	77.71	77.95	77.83	77.31	79.60	77.50	78.82	78.09	
p_A - p_B		6.10	3.49	4.35	5.87	6.19	1.31	4.60	5.10	14.81	3.71	4.9185
ACM												
	Métrica	1	2	3	4	5	6	7	8	9	10	t-valor
SMART *ps*	*macroF1*	70.70	69.10	66.70	67.10	68.30	69.60	71.90	68.10	71.50	68.60	
MetaLazy		64.79	66.72	63.23	63.53	64.39	64.59	65.49	64.80	66.72	66.46	
p_A - p_B		6.21	2.38	3.47	1.57	3.91	5.01	6.41	3.3	4.78	2.14	7.4307
SMART *ps*	*microF1*	79.60	78.90	79.40	79.60	79.50	78.10	80.00	80.30	79.00	78.40	
MetaLazy		78.79	78.86	78.45	78.56	78.87	77.29	78.69	80.01	78.84	78.91	
p_A - p_B		0.81	0.04	0.95	1.04	0.63	0.81	1.31	0.29	0.16	-0.51	3.1794
YELP												
	Métrica	1	2	3	4	5	6	7	8	9	10	t-valor
SMART *ps*	*macroF1*	95.00	94.20	94.40	94.20	94.40	96.60	95.80	94.20	97.00	95.00	
MetaLazy		91.59	92.99	87.98	90.40	90.98	92.79	92.19	91.39	93.60	89.99	
p_A - p_B		3.41	1.21	6.42	3.80	3.42	3.81	3.61	2.81	3.40	5.01	8.6472
SMART *ps*	*microF1*	95.00	94.20	94.40	94.20	94.40	96.60	95.80	94.20	97.00	95.00	
MetaLazy		91.60	93.00	88.00	90.04	91.00	92.80	92.20	91.40	93.60	90.00	
p_A - p_B		3.40	1.20	6.40	4.16	3.40	3.80	3.60	2.80	3.40	5.00	8.6684
STANFORD												
	Métrica	1	2	3	4	5	6	7	8	9	10	t-valor
SMART *ps*	*macroF1*	86.50	78.40	91.70	80.40	83.30	82.90	66.60	77.10	85.70	79.90	
MetaLazy		86.44	83.67	86.10	77.77	83.28	82.85	83.28	82.84	85.71	79.93	
p_A - p_B		0.02	-5.27	5.60	2.63	0.02	0.05	-16.68	-5.74	-0.01	-0.03	0.9992
SMART *ps*	*microF1*	86.50	78.40	91.70	80.60	83.30	83.30	66.70	77.10	87.70	80.00	
MetaLazy		86.48	83.78	86.11	77.77	83.33	83.33	83.33	82.85	85.71	80.00	
p_A - p_B		0.02	-5.38	5.59	2.83	-0.03	-0.03	-16.63	-5.75	1.99	0.00	0.8798

Conforme apresentado na subseção 2.2.4, um *t*-teste pareado pode ser realizado com a condução de *i* execuções dos algortimos comparados. Nos nossos experimentos utilizamos os 10 *folds* de cada conjunto de dados como as *i* execuções. Com os valores da linha $p_A - p_B$, o nível de significância *0.05*, e o grau de liberdade *n - 1 = 9*, podemos calcular o *t*-valor. Com o nível de significância e o grau de liberdade encontramos, na

tabela de distribuição T [1] (two-tails), o valor tabulado de *2.262*. Este valor encontrado, na tabela de distribuição, é comparado com o *t*-valor ($|t|$) calculado para cada conjunto de dados e métrica. A hipótese nula pode ser rejeitada se $|t| > t_{9,0.975} = 2.262$, se $|t| \leq t_{9,0.975} = 2.262$ aceita-se a hipótese nula. Na tabela 3.8, com os valores da coluna *t*-valor, podemos verificar que o *t*-teste pareado indica:

- Hipótese nula rejeitada para o conjunto de dados 20NG. O Método *MetaLazy* possui melhores resultados, em *macroF1* e *microF1*;
- Hipótese nula rejeitada para os conjuntos de dados REUT90, 4UNI, ACM e YELP. Sendo que o método SMART *ps* possui os melhores resultados, tanto em *macroF1* quanto em *microF1*;
- Hipótese nula aceita para o conjunto de dados STANFORD nas métricas *macroF1* e *microF1*.

3.3.3 Versões do método SMART

Inicialmente, o *ensemble* do método proposto, foi construido com a utilização de seis classificadores base (*SGD*, *Logistic Regression*, *SVM*, *Naive Bayes Multinomial*, *Random Forest* e *k-NN*). Com os experimentos realizados foi observado que os métodos *Random Forest* e *k-NN* tem alto custo computacional no momento da predição dos documentos de consulta, além disso o método *Random Forest* também apresenta alto custo computacional no treinamento do classificador. Após avaliar os resultados de precisão e tempo de execução da primeira versão, foi criada uma segunda versão do método com a utilização de apenas quatro classificadores base, nessa segunda versão foram retirados os classificadores *Random Forest* e *k-NN*.

Tabela 3.9: Comparativo da precisão entre as duas versões com 4 e 6 classificadores do método SMART.

		Nº Classif.	20NG	REUT90	4UNI	ACM	YELP	STANFORD
SMART *vm*	macroF1	4	89.68	38.79	73.58	68.37	94.94	81.21
	microF1		89.97	70.06	82.97	79.33	94.94	81.33
SMART *ps*	macroF1	4	89.77	39.72	74.54	69.14	95.00	81.25
	microF1		90.06	70.19	83.02	79.25	95.00	81.33
SMART *pa*	macroF1	4	89.75	39.41	74.52	69.16	95.00	81.25
	microF1		90.03	70.02	83.07	79.10	95.00	81.33
SMART *vm*	macroF1	6	89.70	35.11	72.28	68.31	95.00	81.21
	microF1		89.95	68.33	83.09	79.96	95.00	81.33
SMART *ps*	macroF1	6	90.11	35.92	73.42	69.60	95.08	81.79
	microF1		90.36	69.19	83.55	79.89	95.08	81.87
SMART *pa*	macroF1	6	90.16	35.83	72.96	69.50	95.12	81.48
	microF1		90.41	69.06	83.41	79.96	95.12	81.60

[1] https://www.sjsu.edu/faculty/gerstman/StatPrimer/t-table.pdf

Estas duas versões do método SMART possuem resultados semelhantes de precisão, conforme apresentado na tabela 3.9. No entanto, o tempo de execução, na etapa de teste, da versão com quatro classificadores, obteve um ganho de desempenho, em relação à versão com seis classificadores. Os resultados são apresentados na tabela 3.10 e demonstram, por exemplo, ganho de desempenho de *10x* para o conjunto de dados ACM e ganho de desempenho de *30x* para o conjunto de dados STANFORD, na etapa de consulta (Predição). Este ganho de desempenho deve-se ao fato de termos retirado os dois algoritmos *Random Forest* e *k-NN* do conjunto de classificadores do nosso método e não por alguma estratégia proposta em nosso método.

O tempo de execução, na etapa de consulta (Predição), foi o que motivou a implementação da versão com quatro classificadores, pois a versão com seis classificadores obtinha tempo de execução acima da *baseline* (*MetaLazy*) utilizada. Como os resultados de precisão, entre as versões do método SMART, são muito semelhantes, o tempo de predição dos documentos de consulta foi o que mais contribuiu para optarmos pela versão com quatro classificadores, no momento da comparação com os demais métodos dos experimentos.

Tabela 3.10: Comparativo do tempo médio de execução (segundos), entre as versões do método SMART com 4 e 6 classificadores.

		Nº Classif.	20NG	REUT90	4UNI	ACM	YELP	STANFORD
SMART *vm*	GridSearch + Treinamento	4	13543	3695	2360	4580	1762	164
	Predição		0.0441	0.0350	0.0231	0.0228	0.0124	0.0275
SMART *ps*	GridSearch + Treinamento	4	13543	3695	2360	4580	1762	130
	Predição		0.0410	0.0353	0.0211	0.0205	0.0102	0.0050
SMART *pa*	GridSearch + Treinamento	4	13543	3695	2360	4580	1762	130
	Predição		0.0447	0.0355	0.0228	0.0209	0.0150	0.0303
SMART *vm*	GridSearch + Treinamento	6	16213	5115	3508	6750	2344	255
	Predição		0.2551	0.1766	0.2174	0.2062	0.1859	0.1657
SMART *ps*	GridSearch + Treinamento	6	16213	5115	3508	6750	2344	255
	Predição		0.2536	0.1793	0.2134	0.2078	0.1824	0.1695
SMART *pa*	GridSearch + Treinamento	6	16213	5115	3508	6750	2344	255
	Predição		0.2534	0.1781	0.2152	0.2083	0.1862	0.1690

3.3.4 Região de competência do documento de consulta

A definição de uma região de competência (vizinhança do documento de consulta) é de fundamental importância para métodos de seleção dinâmica do classificador, uma vez que o desempenho das técnicas de seleção dinâmica é muito sensível à definição desta região [Cruz, Sabourin e Cavalcanti 2018]. De acordo com [Cruz et al. 2017], na seleção dinâmica a competência dos classificadores é estimada com base em uma região local do espaço de características, onde está localizado o documento de consulta, denominada região de competência. Essa região é geralmente definida pela aplicação da técnica *k*-NN, para encontrar a vizinhança do documento de consulta. Em seguida, o nível de competência dos classificadores base é estimado, considerando ape-

nas os documentos pertencentes à região de competência de acordo com alguns critérios de seleção. Estes critérios incluem a precisão dos classificadores base nesta região local [Woods, Kegelmeyer e Bowyer 1997] ou ranqueamento [Sabourin et al. 1993] e modelos probabilísticos [Woloszynski et al. 2012]. Em seguida são selecionados o(s) classificador(es) que alcançaram um determinado nível de competência.

Com a realização dos experimentos, também foi avaliado qual o número de vizinhos, do documento de consulta, mais adequado para o nosso método definir a região de competência do classificador. Este valor de k é utilizado na etapa de consulta (teste), para encontrar os vizinhos mais próximos, para cada documento de consulta a ser classificado. Os vizinhos são identificados utilizando o algoritmo k-NN com o uso do cosseno como medida de distância. Com a identificação dos vizinhos, o nosso método utiliza-os para selecionar os classificadores base mais competentes (para esta região de competência), que serão usados para classificar o documento de consulta. Foram feitos experimentos utilizando os seguintes intervalos de valores para k,:

- De 1 a 20, variando de 2 em 2;
- De 1 a 1.000, variando de 100 em 100;
- Valores aleatórios: 10, 20, 30, 50, 70.

Com a avaliação dos resultados obtidos foram escolhidos os seguintes valores para $k = \{5, 10, 100, 200\}$, pois com estes valores nosso métodos alcançou os melhores resultados de precisão. Na tabela 3.11 são apresentados resultados de precisão (medidas *macroF1* e *microF1*), alcançados para diferentes valores de k, utilizando o método SMART com quatro classificadores. Como pode ser observado na tabela 3.11, geralmente os melhores resultados são alcançados para valores pequenos de k. Isto indica que quanto mais próxima é a vizinhança do documento de consulta (região de competência), melhores são os resultados de precisão.

Apesar desses valores de k terem contribuído para o método SMART alcançar resultados significativos, em relação à *baseline*, é necessário um estudo mais aprofundado para encontrar o valor ideal de k, na definição da região de competência do classificador. Os trabalhos [Cruz, Sabourin e Cavalcanti 2018] e [Cruz et al. 2017] apresentam algumas abordagens, além do algoritmo k-NN [Ko, Sabourin e Jr 2008] e [Woods, Kegelmeyer e Bowyer 1997], que podem ser adotadas para definir a região de competência do classificador, como métodos de agrupamento (por exemplo, *K-Means*) [Kuncheva 2000], modelo de função potencial [Woloszynski et al. 2012] e espaço de decisão [Giacinto e Roli 2001]. Um valor de k mais ajustado ao método proposto pode contribuir para ganho de precisão do classificador. Para o conjunto de dados STANFORD só foram utilizados os valores de $k = \{5, 10, 100\}$, na busca dos vizinhos do documento de consulta, porque o conjunto de dados de validação tem menos de 200 documentos.

Tabela 3.11: Resultados de precisão para diferentes valores de k. Métodos com 4 classificadores

	k	Métrica	20NG	REUT90	4UNI	ACM	YELP	STANFORD
SMART *vm*	5	macroF1	89.68	38.79	73.58	68.37	94.94	81.21
		microF1	89.97	70.06	82.97	79.33	94.94	81.33
SMART *ps*	5	macroF1	89.71	39.72	74.04	69.14	95.02	81.25
		microF1	89.99	70.19	82.78	79.25	95.02	81.33
SMART *pa*	5	macroF1	89.70	39.41	74.12	69.14	95.02	81.25
		microF1	89.99	70.02	82.89	79.29	95.02	81.33
SMART *vm*	10	macroF1	89.67	38.54	73.58	68.25	94.94	81.21
		microF1	89.96	70.25	82.96	79.34	94.94	81.33
SMART *ps*	10	macroF1	89.77	39.24	74.52	68.96	95.00	80.97
		microF1	90.06	70.97	83.03	79.32	95.00	81.04
SMART *pa*	10	macroF1	89.75	39.09	74.54	69.05	95.00	80.97
		microF1	90.03	70.80	83.07	79.33	95.00	81.04
SMART *vm*	100	macroF1	89.67	37.77	73.58	68.25	94.94	81.21
		microF1	89.96	70.08	82.96	79.34	94.94	81.33
SMART *ps*	100	macroF1	89.85	39.05	74.57	68.93	95.02	81.22
		microF1	90.01	71.46	83.09	79.29	95.02	81.30
SMART *pa*	100	macroF1	89.83	39.06	74.74	68.94	95.00	81.22
		microF1	90.10	71.50	83.16	79.28	95.00	81.30
SMART *vm*	200	macroF1	89.67	37.77	73.58	68.25	94.94	–
		microF1	89.96	70.08	82.96	79.34	94.94	–
SMART *ps*	200	macroF1	89.83	39.15	74.60	68.92	94.08	–
		microF1	90.09	71.47	83.07	79.24	94.08	–
SMART *pa*	200	macroF1	89.85	39.11	74.57	68.97	95.00	–
		microF1	90.11	71.46	83.07	79.26	95.00	–

3.3.5 Discussões

O presente trabalho defende a criação de um conjunto (*ensemble*) de classificadores para realizar a tarefa de classificação automática de documentos. O método proposto SMART, utiliza diversos tipos de classificadores base (heterogêneos), no intuito de melhorar a precisão final na classificação de documentos. A escolha dos classificadores base foi baseada na precisão e diversidade destes classificadores. Nosso método foi implementado com três formas de *ensemble*, SMART *vm*, SMART *ps* e SMART *pa* e diferencia-se, de outras abordagens da literatura, na maneira de treinar os classificadores base. Realizamos os experimentos em conjuntos de dados padronizados amplamente utilizados na tarefa de ADC.

A avaliação dos resultados mostra ganho de precisão ou resultados muito competitivos tanto em relação ao método *MetaLazy* (*baseline*) quanto em relação aos outros métodos utilizados nos experimentos. As três formas de *ensemble* do nosso método tem resultados muito competitivos em relação à *baseline* utilizada. Apesar dos resultados promissores do nosso método, existem abordagens que podem melhorar o seu desempenho. Na seção 4.3, são indicadas algumas técnicas que podem aprimorar o método propopsto neste trabalho.

O uso de unidades de processamento gráfico (*GPUs*) tem se mostrado promis-

sor na classificação automática de documentos, conforme podemos ver nos trabalhos [Rocha et al. 2015] e [Canuto et al. 2015]. Pretendemos continuar a investigação do uso de paralelismo com *GPUs*, para acelerar a tarefa de classificação automática de documentos, e a avaliação do método proposto em conjuntos de dados maiores, como RCV1 E MEDLINE.

CAPÍTULO 4

Conclusões

4.1 Conclusões

O presente trabalho apresenta como principal contribuição a construção de um classificador automático de documentos do tipo *ensemble*. No decorrer da pesquisa foi obtida a seguinte implementação:

- Método de classificação automática de documentos do tipo *ensemble*, baseado na seleção customizada do classificador base considerando a região local de competência do classificador;

Durante o desenvolvimento do método SMART foram esperimentadas várias abordagens para a construção do conjunto de classificadores e seleção customizada do classificador. Com a realização dos experimentos nossa abordagem final mostrou-se competitiva quando comparada ao método *MetaLazy* (baseline) e também quando comparada a outros métodos conhecidos de classificação automática de documentos. Os experimentos realizados sugerem que o método SMART obteve ganho de precisão ou resultados muito competitivos, tanto em relação ao método *MetaLazy*, quanto em relação aos outros métodos experimentados na tarefa de classificação automática de documentos. A Análise dos tempos de execução demonstra que o método SMART consegue um ganho computacional da ordem de até *5x*, nos conjuntos de dados experimentados, em relação ao método *MetaLazy*. Esse ganho computacional foi alcançado na etapa de consulta (predição).

Apesar dos resultados promissores acreditamos que nossa abordagem pode ser melhorada. Como nossa abordagem é customizada para cada documento de consulta, pode ser realizada a aplicação de técnicas de escolha do classificador ou conjunto de classificadores mais competente, para uma determinada região local de competência do documento de consulta. Atualmente essas técnicas tem demonstrado bons resultados na literatura, como pode ser observado nos trabalhos [Cruz, Cavalcanti e Ren 2011], [Jr, Sabourin e Oliveira 2014], [Cruz, Sabourin e Cavalcanti 2018].

A implementação realizada no trabalho proposto possibilitou a publicação do seguinte artigo:

- Classificação automática de documentos: aplicações e aceleração - ERAD 2020.

4.2 Resultados alcançados

Com a implementação do método proposto foram alcançados os seguintes resultados:

- Seleção customizada do classificador para cada documento de consulta. Ao invés de selecionar um classificador ou um conjunto de classificadores para um determinado conjunto de dados nosso método seleciona o classificador ou o conjunto de classificadores, mais competente, para cada documento de consulta que é apresentado;
- Treinamento dos classificadores base, utilizando os conjuntos de dados de treino e validação, antes da etapa de consulta. Nosso método diferencia-se de outras abordagens encontradas na literatura, pois nestas abordagens o treinamento é realizado utilizando os k vizinhos do documento de consulta como conjunto de dados de treino e na etapa de consulta;
- Ganho de acurácia ou resultados muito competitivos tanto em relação ao método *MetaLazy* (baseline) quanto em relação aos outros métodos utilizados nos experimentos;
- Ganho computacional, de até $5x$, em relação ao método *MetaLazy*, na etapa de consulta (predição);

4.3 Trabalhos futuros

Como contribuições futuras, que podem ser exploradas para dar continuidade e melhorar a proposta apresentada neste trabalho, podem ser relacionados os seguintes pontos:

- Aplicação de técnicas de escolha da região de competência do documento de consulta, onde o nível de competência do classificador ou do *emsemble* é avaliado. Segundo os autores [Cruz, Sabourin e Cavalcanti 2018], essas técnicas podem contribuir para melhorar o resultado final da tarefa de classificação;
- Escolha de múltiplos critérios para avaliar (medir) a competência dos classificadores base, na região local, para alcançar uma técnica mais robusta de seleção dinâmica do classificador;

- Avaliação de outros classificadores para melhorar o desempenho dos resultados obtidos no método SMART;
- Finalização da versão em paralelo na GPU para acelerar a tarefa de Classificação Automática de Documentos;
- Avaliação do nosso método em conjuntos de dados maiores como RCV1 ou Medline para tentar obter maior ganho de precisão.

Referências Bibliográficas

[Aggarwal e Zhai 2012]AGGARWAL, C. C.; ZHAI, C. A survey of text classification algorithms. In: *Mining text data*. [S.l.]: Springer, 2012. p. 163–222.

[Almeida et al. 2012]ALMEIDA, P. R. L. de et al. Music genre classification using dynamic selection of ensemble of classifiers. In: IEEE. *2012 IEEE international conference on systems, man, and cybernetics (SMC)*. [S.l.], 2012. p. 2700–2705.

[Amorin et al. 2018]AMORIN, L. A. et al. A fast similarity search knn for textual datasets. In: SBC. *In Proceedings of the XIX Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 18)*. [S.l.], 2018. p. 421–432.

[Baeza-Yates e Ribeiro-Neto 2013]BAEZA-YATES, R.; RIBEIRO-NETO, B. *Recuperação de Informação-: Conceitos e Tecnologia das Máquinas de Busca*. [S.l.]: Bookman Editora, 2013.

[Bauer e Kohavi 1999]BAUER, E.; KOHAVI, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, Springer, v. 36, n. 1-2, p. 105–139, 1999.

[Bellman 1961]BELLMAN, R. Curse of dimensionality. *Adaptive control processes: a guided tour. Princeton, NJ*, v. 3, p. 2, 1961.

[Bergstra e Bengio 2012]BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, JMLR. org, v. 13, n. 1, p. 281–305, 2012.

[Boser, Guyon e Vapnik 1992]BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: *Proceedings of the fifth annual workshop on Computational learning theory*. [S.l.: s.n.], 1992. p. 144–152.

[Bottou 2010]BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. [S.l.]: Springer, 2010. p. 177–186.

[Breiman 2001]BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001.

[Campos et al. 2017]CAMPOS, R. et al. Stacking bagged and boosted forests for effective automated classification. In: *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. [S.l.: s.n.], 2017. p. 105–114.

[Canuto et al. 2015]CANUTO, S. et al. An efficient and scalable metafeature-based document classification approach based on massively parallel computing. In: ACM. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. [S.l.], 2015. p. 333–342.

[Canuto et al. 2018]CANUTO, S. et al. A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 30, n. 12, p. 2242–2256, 2018.

[Cavalin, Sabourin e Suen 2013]CAVALIN, P. R.; SABOURIN, R.; SUEN, C. Y. Dynamic selection approaches for multiple classifier systems. *Neural computing and applications*, Springer, v. 22, n. 3-4, p. 673–688, 2013.

[Cawley, Talbot e Girolami 2007]CAWLEY, G. C.; TALBOT, N. L.; GIROLAMI, M. Sparse multinomial logistic regression via bayesian l1 regularisation. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2007. p. 209–216.

[Chang e Lin 2011]CHANG, C.-C.; LIN, C.-J. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, Acm New York, NY, USA, v. 2, n. 3, p. 1–27, 2011.

[Chervonenkis e Vapnik 1971]CHERVONENKIS, A.; VAPNIK, V. Theory of uniform convergence of frequencies of events to their probabilities and problems of search for an optimal solution from empirical data(average risk minimization based on empirical data, showing relationship of problem to uniform convergence of averages toward expectation value). *Automation and Remote Control*, v. 32, p. 207–217, 1971.

[Cristianini, Shawe-Taylor et al. 2000]CRISTIANINI, N.; SHAWE-TAYLOR, J. et al. *An introduction to support vector machines and other kernel-based learning methods*. [S.l.]: Cambridge university press, 2000.

[Cruz, Cavalcanti e Ren 2011]CRUZ, R. M.; CAVALCANTI, G. D.; REN, T. I. A method for dynamic ensemble selection based on a filter and an adaptive distance to improve the quality of the regions of competence. In: IEEE. *The 2011 International Joint Conference on Neural Networks*. [S.l.], 2011. p. 1126–1133.

[Cruz, Sabourin e Cavalcanti 2018]CRUZ, R. M.; SABOURIN, R.; CAVALCANTI, G. D. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, Elsevier, v. 41, p. 195–216, 2018.

[Cruz et al. 2015]CRUZ, R. M. et al. Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern recognition*, Elsevier, v. 48, n. 5, p. 1925–1935, 2015.

[Cruz et al. 2017]CRUZ, R. M. et al. Dynamic ensemble selection vs k-nn: why and when dynamic selection obtains higher classification performance? In: IEEE. *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*. [S.l.], 2017. p. 1–6.

[Demšar 2006]DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, v. 7, n. Jan, p. 1–30, 2006.

[Deng et al. 2016]DENG, Z. et al. Efficient knn classification algorithm for big data. *Neurocomputing*, Elsevier, v. 195, p. 143–148, 2016.

[Devlin et al. 2018]DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint:1810.04805*, 2018.

[Dietterich 1998]DIETTERICH, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, MIT Press, v. 10, n. 7, p. 1895–1923, 1998.

[Dong e Han 2004]DONG, Y.-S.; HAN, K.-S. A comparison of several ensemble methods for text categorization. In: IEEE. *IEEE International Conference onServices Computing, 2004.(SCC 2004). Proceedings. 2004*. [S.l.], 2004. p. 419–422.

[Duin 2002]DUIN, R. P. The combining classifier: to train or not to train? In: IEEE. *Object recognition supported by user interaction for service robots*. [S.l.], 2002. v. 2, p. 765–770.

[Fayyad, Piatetsky-Shapiro e Smyth 1996]FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, ACM New York, NY, USA, v. 39, n. 11, p. 27–34, 1996.

[Fernández-Delgado et al. 2014]FERNÁNDEZ-DELGADO, M. et al. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 3133–3181, 2014.

[Genkin, Lewis e Madigan 2007]GENKIN, A.; LEWIS, D. D.; MADIGAN, D. Large-scale bayesian logistic regression for text categorization. *Technometrics*, Taylor & Francis, v. 49, n. 3, p. 291–304, 2007.

[Giacinto e Roli 2001]GIACINTO, G.; ROLI, F. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, Citeseer, v. 34, n. 9, p. 1879–1882, 2001.

[Hossin e Sulaiman 2015]HOSSIN, M.; SULAIMAN, M. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, Academy & Industry Research Collaboration Center (AIRCC), v. 5, n. 2, p. 1, 2015.

[Huang e Suen 1995]HUANG, Y. S.; SUEN, C. Y. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 17, n. 1, p. 90–94, 1995.

[Jahrer, Töscher e Legenstein 2010]JAHRER, M.; TÖSCHER, A.; LEGENSTEIN, R. Combining predictions for accurate recommender systems. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2010. p. 693–702.

[Jiménez, Lázaro e Dorronsoro 2009]JIMÉNEZ, Á. B.; LÁZARO, J. L.; DORRONSORO, J. R. Finding optimal model parameters by deterministic and annealed focused grid search. *Neurocomputing*, Elsevier, v. 72, n. 13-15, p. 2824–2832, 2009.

[Jr, Sabourin e Oliveira 2014]JR, A. S. B.; SABOURIN, R.; OLIVEIRA, L. E. Dynamic selection of classifiers—a comprehensive review. *Pattern recognition*, Elsevier, v. 47, n. 11, p. 3665–3680, 2014.

[Ko, Sabourin e Jr 2008]KO, A. H.; SABOURIN, R.; JR, A. S. B. From dynamic classifier selection to dynamic ensemble selection. *Pattern recognition*, Elsevier, v. 41, n. 5, p. 1718–1731, 2008.

[Krstajic et al. 2014]KRSTAJIC, D. et al. Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of cheminformatics*, BioMed Central, v. 6, n. 1, p. 1–15, 2014.

[Kuang e Zhao 2009]KUANG, Q.; ZHAO, L. A practical gpu based knn algorithm. In: CITESEER. *Proceedings. The 2009 International Symposium on Computer Science and Computational Technology (ISCSCI 2009)*. [S.l.], 2009. p. 151.

[Kuncheva 2000]KUNCHEVA, L. I. Clustering-and-selection model for classifier combination. In: IEEE. *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No. 00TH8516)*. [S.l.], 2000. v. 1, p. 185–188.

[Kuncheva 2002]KUNCHEVA, L. I. A theoretical study on six classifier fusion strategies. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 24, n. 2, p. 281–286, 2002.

[Kuncheva 2014]KUNCHEVA, L. I. *Combining pattern classifiers: methods and algorithms*. [S.l.]: John Wiley & Sons, 2014.

[Kuncheva e Whitaker 2003]KUNCHEVA, L. I.; WHITAKER, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, Springer, v. 51, n. 2, p. 181–207, 2003.

[Larochelle et al. 2007]LAROCHELLE, H. et al. An empirical evaluation of deep architectures on problems with many factors of variation. In: *Proceedings of the 24th international conference on Machine learning*. [S.l.: s.n.], 2007. p. 473–480.

[LeCun et al. 1998]LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

[Lyman e Varian 2004]LYMAN, P.; VARIAN, H. How much information 2003? 2004.

[Manning, Raghavan e Schütze 2010]MANNING, C.; RAGHAVAN, P.; SCHÜTZE, H. Introduction to information retrieval. *Natural Language Engineering*, Cambridge university press, v. 16, n. 1, p. 100–103, 2010.

[Mendes L. F. 2020]MENDES L. F., e. a. "keep it simple, lazy"– metalazy: a new metastrategy for lazy text classification. *Conference on Information and Knowledge Management – CIKM*, 2020.

[Mitchell et al. 1997]MITCHELL, T. M. et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, v. 45, n. 37, p. 870–877, 1997.

[Nair e Hinton 2010]NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *ICML*. [S.l.: s.n.], 2010.

[Nguyen et al. 2016]NGUYEN, T. T. et al. A novel combining classifier method based on variational inference. *Pattern Recognition*, Elsevier, v. 49, p. 198–212, 2016.

[Pereira, Mitchell e Botvinick 2009]PEREIRA, F.; MITCHELL, T.; BOTVINICK, M. Machine learning classifiers and fmri: a tutorial overview. *Neuroimage*, Elsevier, v. 45, n. 1, p. S199–S209, 2009.

[Polikar 2006]POLIKAR, R. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, IEEE, v. 6, n. 3, p. 21–45, 2006.

[Prasetijo et al. 2017]PRASETIJO, A. B. et al. Hoax detection system on indonesian news sites based on text classification using svm and sgd. In: IEEE. *2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. [S.l.], 2017. p. 45–49.

[Press et al. 1992]PRESS, W. H. et al. *Numerical recipes in Fortran 77: volume 1, volume 1 of Fortran numerical recipes: the art of scientific computing*. [S.l.]: Cambridge university press, 1992.

[Ren, Zhang e Suganthan 2016]REN, Y.; ZHANG, L.; SUGANTHAN, P. N. Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational intelligence magazine*, IEEE, v. 11, n. 1, p. 41–53, 2016.

[Rocha et al. 2015]ROCHA, L. et al. G-knn: an efficient document classification algorithm for sparse datasets on gpus using knn. In: ACM. *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. [S.l.], 2015. p. 1335–1338.

[Rokach 2009]ROKACH, L. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational statistics & data analysis*, Elsevier, v. 53, n. 12, p. 4046–4072, 2009.

[Sabourin et al. 1993]SABOURIN, M. et al. Classifier combination for hand-printed digit recognition. In: IEEE. *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*. [S.l.], 1993. p. 163–166.

[Santos, Sabourin e Maupin 2008]SANTOS, E. M. D.; SABOURIN, R.; MAUPIN, P. A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern recognition*, Elsevier, v. 41, n. 10, p. 2993–3009, 2008.

[Sarkar 2016]SARKAR, D. Text analytics with python. Springer, 2016.

[Sebastiani 2002]SEBASTIANI, F. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 34, n. 1, p. 1–47, 2002.

[Stone 1974]STONE, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, Wiley Online Library, v. 36, n. 2, p. 111–133, 1974.

[Woloszynski et al. 2012]WOLOSZYNSKI, T. et al. A measure of competence based on random classification for dynamic ensemble selection. *Information Fusion*, Elsevier, v. 13, n. 3, p. 207–213, 2012.

[Wolpert 2002]WOLPERT, D. H. The supervised learning no-free-lunch theorems. In: *Soft computing and industry*. [S.l.]: Springer, 2002. p. 25–42.

[Woods, Kegelmeyer e Bowyer 1997]WOODS, K.; KEGELMEYER, W. P.; BOWYER, K. Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 19, n. 4, p. 405–410, 1997.

[Woźniak, Graña e Corchado 2014]WOŹNIAK, M.; GRAÑA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, Elsevier, v. 16, p. 3–17, 2014.

[Zhu, Wu e Yang 2004]ZHU, X.; WU, X.; YANG, Y. Dynamic classifier selection for effective mining from noisy data streams. In: IEEE. *Fourth IEEE International Conference on Data Mining (ICDM'04)*. [S.l.], 2004. p. 305–312.