UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Welton Augusto Rodrigues Santos

PTFS - Previsão e Tratamento de Falhas em modelos de Stacking

Belo Horizonte
2024

Welton Augusto Rodrigues Santos

PTFS - Previsão e Tratamento de Falhas em modelos de Stacking

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Marcos André Gonçalves
Coorientador: Leonardo Chaves Dutra da Rocha

Belo Horizonte
2024

Santos, Welton Augusto Rodrigues Santos.

S237p PTFS - Previsão e Tratamento de Falhas em modelos de Stacking [recurso eletrônico] / Welton Augusto Rodrigues Santos– 2024.
1 recurso online (63 f. il., color.) : pdf.

Orientador: Marcos André Gonçalves.
Coorientador: Leonardo Chaves Dutra da Rocha.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 57-63

1. Computação – Teses. 2. Aprendizado do computador – Teses. 3. Processamento de linguagem natural (Computação) – Teses. 4. Classificação (Computadores) -– Teses. I. Gonçalves, Marcos André. II. Rocha, Leonardo Chaves Dutra da.
III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. IV. Título.

CDU 519.6*82.10(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

PTFS - Previsão e Tratamento de Falhas em modelos de Stacking

WELTON AUGUSTO RODRIGUES SANTOS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Marcos André Gonçalves - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Leonardo Chaves Dutra da Rocha - Coorientador
Departamento de Ciência da Computação - UFSJ

Prof. Sérgio Daniel Carvalho Canuto
Departamento de Ciência da Computação - IFG

Prof. Thierson Couto Rosa
Instituto de Informática - UFG

Belo Horizonte, 3 de junho de 2024.

Dedicuum cest laborae a quelquis personatum que ajudorat a facirelo.

Agradecimentos

Primeiramente agradeço a Deus por me permitir realizar este trabalho e me sustentar perante todas as dificuldades encontradas ao longo do caminho. Agradeço aos meus pais por serem o porto seguro da minha vida e me apoiarem em minhas decisões. Agradeço aos meus orientadores por me guiarem neste trabalho e facilitarem o processo de evolução pessoal ao longo da caminhada. Agradeço a minha namorada por todo amor, compreensão e apoio a mim dedicados. Agradeço a minha terapeuta por me ajudar nos inúmeros momentos de aflição deparados pelo caminho. Agradeço aos inúmeros amigos que me ajudaram diretamente e indiretamente no desenvolvimento desta sociedade anônima de contribuintes indispensáveis que é minha dissertação.

"*I have to go far, because alone is never an option.*"
(Otsugua Notlew)

Resumo

Modelos de empilhamento (*stacking*) são amplamente utilizados em Classificação Automática de Documentos (CAD). Empilhamento combina diferentes classificadores (modelos base) através de um meta-classificador que explora as diferentes habilidades e complementaridades dos modelos base para alavancar a efetividade na classificação. Porém, em abordagens tradicionais de *stacking*, o meta-classificador é limitado a aprender apenas uma única combinação de modelos base para todas instâncias em uma base de dados. Esta limitação prejudica o desempenho do *stacking* pois, quando um modelo base falha (ou possui uma alta chance de falhar) em classificar corretamente um documento (teste), sua importância para decisão final do meta classificador deveria ser reduzida. Isso porém não ocorre nas soluções correntes.

Neste trabalho, nos concentramos em tratar essa limitação do meta-classificador em lidar com potenciais falhas de modelos base. Mais especificamente, estamos interessados em desenvolver estratégias para auxiliar o meta-classificador a ajustar a importância dos modelos base de acordo com a expectativa de sucesso de cada modelo para para cada documento em específico. Nosso trabalho busca responder questões de pesquisa tais como: (i) É possível atingir o máximo do potencial do *stacking* prevendo e tratando falhas dos modelos base? (ii) É possível encontrar o modelo base mais adequado para classificar um documento em específico? Para responder tais perguntas, apresentamos o *arcabouço* **Previsão e Tratamento de falhas em modelos de Stacking** (PTFS). Nosso *arcabouço* comporta três estratégias (Error Detection, Best Model e Hard Docs) voltadas para identificar e tratar falhas dos modelos base, potencialmente reduzindo o impacto dos insucessos desses modelos no desempenho do final do meta-classificador e do *stacking* no geral.

Apresentamos uma ampla avaliação das estratégias contidas no PTFS utilizando múltiplas bases de dados e com diversos classificadores (modelos) base. Apesar das respostas negativas para várias das perguntas de pesquisa, nosso estudo contribui com interessantes percepções e análises que podem guiar trabalhos futuros.

Palavras-chave: processamento natural de linguagem; stacking; previsão de falhas; tratamento de falhas; seleção dinâmica em ensembles.

Abstract

Stacking models are widely used in Automatic Text Classification (ATC). Stacking combines different classifiers (base models) through a meta-classifier that exploits the different abilities and complementarities of the base models to leverage effectiveness in classification. However, in traditional stacking approaches, the meta-classifier is limited to learning only a single combination of base models for all instances in a database. This limitation hampers the performance of stacking because, when a base model fails (or has a high chance of failing) to correctly classify a document (test), its importance for the final decision of the meta-classifier should be reduced. However, this does not occur in current solutions.

In this work, we focus on addressing this limitation of the meta-classifier in dealing with potential failures of base models. More specifically, we are interested in developing strategies to assist the meta-classifier in adjusting the importance of the base models according to the expected success of each model for each specific document. Our work aims to answer research questions such as: (i) Is it possible to achieve the maximum potential of stacking by predicting and addressing failures of base models? (ii) Is it possible to find the most suitable base model to classify a specific document? To answer such questions, we present framework **Prediction and Treatment of failures in Stacking models** (PTFS). Our framework encompasses three strategies (Error Detection, Best Model, and Hard Docs) aimed at identifying and addressing failures of base models, potentially reducing the impact of these models' failures on the performance of the final meta-classifier and stacking overall.

We present a comprehensive evaluation of the strategies contained in PTFS using multiple databases and various base classifiers (models). Despite negative answers to several research questions, our study provides interesting insights and analyses that can guide future work.

Keywords: natural language processing; stacking; failures prediction; failures treatment; dynamic ensemble selection.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

20NG	*20 Newsgroup*
ACM	*ACM Digital Library*
AGNews	*AG's News*
BMD	*Best Model Detection*
BOW	*Bag-of-Words*
CAD	Classificação Automática de Documentos
DD	Detector de documentos Difíceis
DES	Dynamic Ensemble Selection
EF	Estimador de Falhas
IMDd	*Internet Movie Dataset*
kNN	*k-Nearest Neighbors*
LR	*Logistic Regression*
MF	Meta-Features
PTFS	Previsão e Tratamento de Falhas em *Stacking*
RF	*Random Forest*
Sogou	*Sogou News Dataset*
SVM	*Support Vector Machines*
TF-IDF	*Term Frequency - Inverse Document Frequency*
Upperbound	Limite Superior do Stacking
WebKB	*World Wide Knowledge Base*

Sumário

Capítulo 1

Introdução

Classificação Automática de Documentos (CAD) é uma das principais tarefas de processamento natural de linguagem. O objetivo desta tarefa é assinalar rótulos a documentos de forma automática com a utilização de estratégias de aprendizado de máquina. Esta tarefa tem aplicação em diversas áreas como detecção de *spam*, análise de sentimentos em *reviews* de produtos e categorização de notícias e artigos de mídia. Atualmente, arquiteturas conhecidas como *transformers* apresentam resultados estado-da-arte na classificação de documentos [24, 43, 64]. No entanto, tais arquiteturas possuem desvantagens como a dificuldade em obter modelos efetivos frente a escassez de dados [62], dificuldade de generalização para diferentes idiomas [51] e elevado custo computacional tanto para pré-treino quanto para fine-tuning destes modelos[22, 59].

Uma das formas de superar as limitações inerentes as arquiteturas *transformers* consiste em combinar estas com diferentes e complementares modelos de CAD. Na literatura abordagens que combinam múltiplos modelos são conhecidas genericamente como *ensembles* [28]. Dentre as diferentes estratégias *ensemble* presentes na literatura, abordagens com base em *stacking* tem apresentado resultados promissores em CAD [30, 27]. *Stacking* consiste em uma estratégia que combina a saída (i.e. predições de múltiplos classificadores heterogêneos - modelos base) como entrada para outro classificador (meta-classificador) com intuito de aumentar a efetividade na classificação [63]. A premissa desta estratégia é que diferentes modelos e diferentes representações de documentos são complementares entre si, de modo que a limitação de um modelo base específico é compensada pela proficiência dos demais. Ao treinar o meta-classificador sobre a saída dos modelos base o meta-classificador aprende a extrair o potencial intrínseco existente na complementariedade dos diferentes modelos base.

Apesar do potencial existente na combinação de modelos pelo *stacking*, estratégias de *stacking* tradicionais possuem a desvantagem de lidar somente com as predições dos modelos base no treinamento do meta-classificador. Durante o treino do meta-classificador, este aprende a combinar as predições dos modelos base com objetivo de maximizar a efetividade geral, criando uma única "regra" utilizada para ponderar as predições dos modelos base para toda e qualquer instancia. Esta característica abre espaço para problemas como sub e super valorização de modelos base. Por exemplo, o meta-classificador

pode enviesar seu aprendizado e dar demasiada importância as predições de um modelo base mais efetivo do *stacking* (devido ao sua superior efetividade em relação aos demais) ao ponto de priorizar suas predições em relação aos demais sobre instancias onde o modelo mais efetivo não é o mais adequado. Resumidamente, este comportamento permite que predições erradas recebam demasiada importância e predições corretas recebam pouca ou nenhuma, ocasionando a falha do meta-classificador.

Trabalhos recentes apresentam abordagens para otimizar o ponderamento de predições provindas dos modelos base com granularidade a nível de instancia [47, 42, 41, 35, 21]. Estas abordagens utilizam diferentes recursos para selecionar conjuntos específicos de classificadores para cada instância e mostram fortes evidências que ponderar a importância dos modelos base a nível de instancia pode aprimorar o desempenho dos *Ensembles*. No entanto, poucos trabalhos exploram a utilização de estimadores de desempenho de modelos para seleção de modelos base para *stacking* a nível de instância. Essa dissertação procura atacar essa lacuna da literatura.

1.1 Objetivos

Em maiores detalhes, o objetivo deste trabalho é entender e tratar o impacto causado pelos equívocos do meta-classificador ao estimar erroneamente a importância dos modelos base do *stacking*. Apresentamos um novo *framework* denominado Previsão e Tratamento de Falhas em *Stacking* (PTFS) que comporta três abordagens para auxiliar o meta-classificador na tarefa de CAD baseadas na estimativa do desempenho (previsão e tratamento de falhas) dos modelos base do *stacking*. São elas: **Error Detection**, **Best Model** e **Hard Docs Detection**. Cada uma delas será detalhada na seção seguinte. Para construir o arcabouço experimental deste trabalho buscamos inspiração em trabalhos prévios da literatura [30].

Os resultados obtidos neste trabalho oferecem insumo para novas abordagens voltadas para *stacking* com foco em estimar falhas em modelos. Além disso, apresentamos novas estratégias de representação de instâncias (que levam em consideração não somente o texto dos documentos) para *ensembles* que podem ser estendidas para diferentes tarefas (e.g., regressão, recuperação de informação) com prática aplicabilidade na previsão de falhas de modelos, beneficiando estratégias de *Automated Machine Learning* i.e., *AutoML*.

1.2 Contribuições

Como *primeira contribuição* deste trabalhos identificamos o Limite Superior do *Stacking* (*Upperbound*) descartando predições errôneas dos modelos base do *stacking*. Deste modo simulamos um meta-classificador ideal. Conhecido o *Upperbound* do *stacking*, investigamos a primeira questão de pesquisa deste trabalho:

QP1 - É possível aproximar a efetividade do Upperbound *do* stacking *através da previsão da falha dos modelos base?*

Para responder QP1 propomos nossa primeira abordagem, denominada **Error Detection**. Nesta abordagem, para cada modelo base do *stacking* treinamos um estimador de erro que consiste em um classificador que possui a função de prever a falha do modelo base. Para tal investigação conduzimos séries de experimentos comparativos com modelos de *stacking* compostos de algoritmos estado da arte em CAD sobre seis base de dados (utilizadas ao longo de todo trabalho) amplamente empregadas como *benchmark* na literatura.

Como *segunda contribuição* introduzimos um novo conjunto de Meta-Features (MFs) descritivas combinando informação conjunta e individual dos modelos base do *stakcing* somado à informação da base de dados para treinar os estimadores de erro.

Como *terceira contribuição* deste trabalho apresentamos uma variação da forma de abordagem utilizada para responder QP1 com foco na simplificação do problema. Esta abordagem é guiada pela segunda questão de pesquisa deste trabalho:

QP2 - É possível aproximar o Upperbound *do* stacking *identificando o melhor modelo base para cada documento?*

Nesta perspectiva, temos o objetivo de facilitar a aproximação do *Upperbound* substituindo a necessidade de prever com elevada assertividade as falhas de múltiplos modelos base em prol da identificação do melhor modelo para o documento. Para responder QP2 treinamos um novo classificador utilizando as MFs empregadas em QP1 com foco na seleção do melhor modelo para cada documento. Denominamos esta estratégia de **Best Model**.

Na *quarta contribuição* deste trabalho mudamos o foco dos esforços empregados na previsão e contenção da falha dos modelos base do *stacking* para construção de meta-classificadores mais robustos contra predições errôneas dos modelos base. Observamos que grande parte das falhas do meta-classificador ocorrem quando a maioria dos modelos base falham (e.g., em um *stacking* composto de sete modelos base, o meta-classificador tende a falhar em documentos onde quatro ou mais modelos base falham). Nesta linha apresentamos a terceira questão de pesquisa deste trabalho:

QP3 - É possível treinar meta-classificadores capazes de predizer corretamente a classe de documentos onde a maioria dos modelos base falham?

Guiado pela QP3 introduzimos a terceira abordagem deste trabalho denominada

Hard Docs Detection. Nesta abordagem estabelecemos um pipeline de dois passos. No primeiro detectamos o que chamamos de documentos difíceis (documentos onde mais de 50% dos modelos base do *stacking* erram). Para isso treinamos um classificador (detector de difíceis) com as MFs previamente descritas. No segundo passo treinamos dois meta-classificadores, um tradicional e outro especializado na predição de documentos difíceis. Ao chegar novos documentos o detector de documentos difíceis decide qual meta-classificador deve fazer a predição.

Como mostrado nos experimentos deste trabalho, na maioria das bases de dados as estratégias propostas possuem um potencial signifciativo. Apesar das estratégias não superarem a eficácia dos modelos de *stacking* tradicionais, apresentam resultados próximos e com grande espaço e flexibilidade para melhorias. Além disso, nossos resultados mostram proeminente efetividade na estimativa de desempenho dos modelos base do *stacking*, mostrando que o uso de informação compartilhada entre diferentes modelos base combinado com informações da base de dados contribuem significativamente para o aumento da assertividade na estimativa do desempenho de classificadores. Tal comportamento nos permite abrir novos caminhos para produção de novas MFs mais capazes de fornecer subsídio para o desenvolvimento de estimadores de desempenho mais efetivos com potencial benefício para sistemas de *AutoML*.

Os resultados desta dissertação geraram uma publicação [57] no 38º anual Simpósio Brasileiro de Bancos de Dados.

1.3 Organização

Esta dissertação está organizada como se segue: Capítulo 2 dispomos de trabalhos base para a compreensão do *stacking*. No capítulo 3 decorremos por trabalhos recentes fonte de inspiração para o corrente trabalho. No capítulo 4 apresentamos a metodologia utilizada para estruturar os experimentos utilizados para responder as questões de pesquisa desta dissertação. No capítulo 5 descrevemos o arcabouço experimental utilizado para execução dos experimentos. No capítulo 6 apresentamos os resultados obtidos respondendo as questões de pesquisa deste trabalho. Por final, no capítulo 7 concluímos o trabalho e indicamos frentes para estudos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo discorremos sobre trabalhos base para o desenvolvimentos de modelos de *stacking* estudados neste trabalho. Na seção 2.1 apresentamos as principais estratégias de representação de documentos. Na seção 2.2 descoremos sobre estratégias de aprendizado supervisionado presentes na literatura. Na seção 2.3 abordamos trabalhos voltados para o estudo de *ensembles*, principalmente *stackings*.

2.1 Representações

Uma das formas mais comuns de representação de documentos na literatura consiste em representar documentos como conjuntos de palavras Bag-of-Words (BOW). Dado uma base de dados de documentos definida como $D = \{d_1, ..., d_n\}$ e $V = \{w_1, ..., w_m\}$ definido como o conjunto de palavras contidos nos documentos em D. Estratégias da família BOW representam cada documento $d_i \in D$ como um vetor $r_i^{|V|}$. Cada dimensão de $r_i^{|V|}$ representa a relação do documento d_i com uma palavra $w_j \in V$. Uma das variantes BOW mais amplamente utilizada na literatura consiste no TF-IDF (*Term Frequency - Inverse Document Frequency*)[56]. TF-IDF representa um documento através da suas palavras ponderando-as de acordo com a sua frequência no documento e o inverso da sua frequência na coletânea completa da base de dados. Logo, para um documento as palavras que melhor o representam tendem a possuir maior peso, uma vez que estas palavras tendem a ser raras na base de dados (aparecem em poucos documentos - baixo IDF), porém possuem elevada frequência no documento (elevado TF).

Um das desvantagens do TF-IDF consiste em não considerar a informação de coocorrência entre palavras, característica que impede a utilização de informação semântica entre palavras. Uma das formas de contornar esta limitação é aplicar o uso de *n-grams* durante a construção do vocabulário, considerando sequencias de palavras que coocorrem frequentemente como novos atributos [18]. No entanto, para sequências de n-grams maiores o custo desta técnica se torna um problema devido ao demasiado crescimento do

número de atributos. Para superar esta limitação estratégias de *embeddings* [45, 50, 9] buscam estimar a distribuição de probabilidade da ocorrência de pares de palavras analisando sua coocorrência em grandes volumes de texto. Estas estratégias projetam palavras semanticamente relacionadas em regiões próximas de em um espaço vetorial, representando palavras como vetores densos (geralmente com 50 a 2000 dimensões).

Outra forma de representar documentos é a partir de Meta-Features. Trabalhos com Meta-Features baseadas em distância tem apresentado promissores resultados [16, 13]. Estas Meta-Features representam documentos pela sua distância em relação aos centroides das classes e de seus vizinhos mais próximos. Geralmente métricas como distância euclidiana e distância de cosseno são utilizadas para combinar a direção e distância espacial entre documentos.

Neste trabalho utilizamos as estratégias TF-IDF e Meta-Features como estratégia de representação de documentos combinadas com diferentes algoritmos de classificação empregados no *stacking*.

2.2 Algoritmos de Classificação

Vários algoritmos de aprendizado supervisionado são empregados em CAD combinados com diferentes representações [36, 39]. Neste trabalho, consideramos diferentes modelos para compor os modelos de *stacking* (e.g., *Support Vector Machines* (SVM), *k-Nearest Neighbors* (kNN) e *Logistic Regression* (LR)) e como base para as estratégias de previsão e tratamento de falha avaliadas ao longo deste estudo utilizamos o algoritmo *Random Forest* (RF).

SVM [20] é um algoritmo que se tornou estado da arte em diferentes áreas do aprendizado supervisionado, mas principalmente em CAD. O objetivo deste algoritmo é encontrar um hiperplano ótimo que separe duas classes em um espaço vetorial N-dimensional. Originalmente é uma estratégia linear, porém, através da estratégia *kernel trick* [10] pode utilizar diferentes funções de *kernel* para realizar transformações não lineares nos dados originais. Apesar de eficiente, para grandes volumes de dados o treino do SVM pode ser demasiadamente caro computacionalmente e em relação a tempo. Em contra partida, estratégias como Regressão Logística apresentam resultados proeminentes com baixo custo computacional [53, 40]. Regressão Logística assim como SVM trata-se de um algoritmo de classificação binária que pode ser estendido para problemas classificação com múltiplas classes. Seu objetivo é aprender a separar dados binários através do ajuste da função logística $p_i = \frac{1}{1+e^{-(\beta_0+\sum_1^k X_{k,i}\beta_k)}}$, onde p_i representa a probabilidade de sucesso de um evento (0 ou 1), X_i são os atributos (preditores) da instância i e β_i são parâmetros

aprendidos durante o treino (β_0 é o coeficiente linear).

kNN consiste em uma estratégia de aprendizado supervisionado que classifica objetos de acordo com seus k vizinhos mais próximos [7]. Esta estratégia apresenta resultados proeminentes em CAD [39], porém, para grandes conjuntos de dados demanda elevada memória e processamento, sendo a superação de tais limitações foco de estudo de diferentes trabalhos [54].

Recentemente modelos estratégias conhecidas com *transformers* [24, 64, 43] tem apresentado resultados estado-da-arte em diferentes vertentes de **Processamento Natural de Linguagem**, mas principalmente em CAD. Estas arquitetura são pré-treinadas em grandes volumes de texto e contam com mecanismo de atenção para capturar informação contextual em documentos [4, 61] podendo ser especializadas posteriormente em tarefas finalísticas a partir do *fine-tuning* [59]. No entanto, essas estratégias necessitam de grande poder computacional provido por Processadores Gráficos (*Graphics Processing Unit*) o que pode ser um limitante no uso de tais estratégias [22].

A diferença entre algoritmos de aprendizado supervisionado é um fator de significativo na diferença de desempenho entre modelos nos diferentes cenários de CAD. Combinar diferentes algoritmos com diferentes representações permite criar estratégias mais robustas em diferentes cenários. Neste trabalho avaliamos a combinação de diferentes algoritmos e representações por meio da estratégia de *stacking*.

2.3 Stacking

Stacking consistem em uma estratégia da família de *ensembles* que combina classificadores heterogêneos (i.e., modelos base) com objetivo de se beneficiar da complementariedade entre os diferentes modelos em prol do aumento da efetividade em tarefas de classificação [63]. A implementação do *stacking* é dividida em duas partes: treinamento dos modelos base e treinamento do meta-classificador. No treinamento dos modelos base, deve-se treinar cada modelo da seguinte forma: no conjunto de treino realiza-se uma validação cruzada para gerar distribuições de probabilidades (predições) que representem cada instância para cada modelo base. Em seguida treina-se o modelo base sobre o conjunto completo de treino e gera-se predições sobre o conjunto de teste (probabilidades dos documentos de teste). No final do processo deve-se ter o conjunto de treino e teste representado por distribuições de probabilidades. No treinamento do meta-classificador as predições dos modelos base são concatenadas e inseridas junto com os rótulos originais dos dados no meta-classificador que aprende a combinar as predições dos modelos base.

Recentemente vários trabalhos tem empregado estratégias de *stacking* com ob-

jetivo de aumentar a efetividade em CAD [44, 12, 46, 17, 1, 30]. Mienye et al. [44] realizam uma revisão sobre trabalhos voltados para estratégias com base em *ensemble* i.e., *boosting*, *bagging* e *stacking*. Os autores abordam trabalhos sobre diferentes áreas (eg., classificação de texto, detecção de doenças, análise de sentimentos). Estratégias de *ensemble* tem sido amplamente utilizadas para melhorar a efetividade em tarefas de CAD e similares (e.g., análise de sentimentos, detecção de conteúdo hostil). Campos et al. [12] combinam estratégias de *boosting*, *bagging* e *stacking* e introduzem um novo algoritmo com base em *Extremely Randomized Trees* denominado BERT. Os autores reportam resultados estado-da-arte em 15 bases de dados cobrindo CAD e analise de sentimentos. Em alguns cenários apresentam ganhos de 20% em relação as demais *baselines*. Além disso apresentam um estudo sobre o tempo necessário para execução do *stacking* o que pode ser um limitante devido ao elevado custo computacional. Inspirado em *stackings* de modelos profundos Mohammed et al. [46] apresentam uma nova estratégia de *stacking* em três níveis. No primeiro nível (base do *stacking*) os autores treinam diferentes modelos profundos (e.g., CNNs, BiLSTMs). No segundo nível os autores utilizam estratégias tradicionais (e.g., SVMs, Regressão Logística) para gerar meta-classificadores primários treinados sobre predições dos modelos base. Por final, parte das predições dos meta-classificadores primários são utilizadas para treinar uma rede neural de camada única que fara a predição final. Os autores reportam resultados superiores aos demais *baselines* em diferentes bases de dados. No entanto, devido aos vários níveis da estratégia para bases pequenas as várias partições nos dados pode ser um problema reduzindo demasiadamente a representação original dos dados no treino e para bases maiores o custo da arquitetura é demasiadamente elevado.

Além do foco na heterogeneidade entre algoritmos de classificação trabalhos recentes mostram resultados esta-da-arte através da combinação de diferentes representações de documentos. Carvalho et al. [17] combinam diferentes formas de representação de documentos (e.g., *n-grams, word embeddings*). Os autores compraram o uso das representações concatenadas e por meio de *stacking*. De 22 bases de dados utilizadas no estudo em 12 bases o *stacking* obteve melhores resultados que a simples concatenação. Abarna et al. [1] introduzem no novas estratégias de *ensemble* (incluindo *stacking*) para classificação de frases idiomáticas. Os autores combinam modelos pré-treinados (e.g., BERT [24], RoBERTa [43]) com uma nova estratégia com base em grafos de conhecimento (*Knowledge Graph-BERT*) combinando informação prévia de frases idiomáticas com modelos pré-treinados. Os autores comparam as estratégias de *ensemble* com modelos individuais e reportam melhorias de efetividade na detecção de frases idiomáticas. Gomes et al. [30] apresentam um extensivo estudo sobre *stacking* com foco na eficiência e custo benefício da técnica. Os autores combinam várias representações com diferentes algoritmos de classificação (envolvendo também modelos de aprendizado profundos) e utilizam oito base de dados amplamente utilizadas como *benchmark* na literatura para avaliação experimental

em CAD. Experimentalmente os autores avaliam a relação eficiência e eficácia considerando duas frentes: volume de dados e combinação de modelos. Na perspectiva do volume de dados, os autores atingem resultados proeminentes com *stacking* utilizando pequenas frações dos conjuntos de dados originais (e.g, 15%, 30% do conjunto de treino). Neste trabalho buscamos inspiração em [30] para construção do arcabouço experimental desta dissertação. Dado a grande diversidade existentes nos modelos de *stacking* apresentados em [30] utilizamos tais modelos para avaliar o impacto das novas estratégias de seleção de *ensemble* propostas neste estudo.

Capítulo 3

Trabalhos Relacionados

Neste capítulo abordamos trabalhos que serviram de inspiração para o trabalho corrente. Na seção 3.1 apresentamos trabalhos com foco no avanço de modelos de *ensemble* através do aumento da diversidade entre modelos base do *stacking*. Na seção 3.2 apresentamos trabalhos com foco na seleção dinâmica de modelos em *ensembles* (seleção de diferentes conjuntos de modelos em um *ensemble* para cada documento).

3.1 Diversidade em Ensembles

Diversidade entre modelos base de um *ensemble* é fundamental para sua efetividade [11, 26], assim como a seleção adequada dos modelos que compõem o ensemble (esparsidade)[67]. A otimização de ensembles com base em diversidade e esparsidade é foco de diferentes trabalhos [55, 38, 65, 32]. Sabzevari et al. [55] apresentam uma estratégia de *weighted boosting* com foco na diversidade dos modelos base. Nesta abordagem os autores utilizam a diversidade entre modelos base para definir o peso das instâncias. Instâncias com maior diversidade recebem maior atenção (peso), com esta estratégia os autores reduzem a incerteza do *ensemble*. Lacoste et al. [38] introduzem uma nova abordagem para seleção de modelos em *ensembles* denominada *agnostic Bayes framework*. O foco da estratégia é modelar a relação entre o risco das hipóteses (modelos base) e o erro empírico observado no desempenho dos modelos base. Os autores utilizam inferência Bayesiana para encontrar o melhor modelo i.e., hipótese com menor risco entre um conjunto de modelos. Avançando as estratégias de seleção de models previamente citadas Yang et al. [65] apresentam uma nova abordagem para seleção de modelos combinando acurácia, diversidade e esparsidade. Os autores apresentam uma nova métrica para medir a diversidade em um *ensemble* e utilizam esta métrica em uma nova função de perda que também considera a acurácia e esparsidade. Guo et al. [32] realizam um estudo teórico sobre os conceitos de *ensemble margins* [5] e *pairwise diversity* [33] e apresentam uma nova abordagem para selecionar modelos em um *ensemble* com base em uma nova

métrica denominada MDM. A métrica usa o conceito de *example margins* que identifica instâncias onde ocorrem maior divergência e menor nível de confiança no *ensemble*. Os autores priorizam tais instâncias para estimar a importância dos modelos do *ensemble* em termos de diversidade e assertividade.

3.2 Seleção Dinâmica de Modelos

Uma limitação presente nas abordagens para seleção de modelos em *ensembles* apresentadas previamente consiste na seleção estática de modelos. Basicamente, após a escolha de quais modelos devem compor o *ensemble*, está escolha será a final para todas as instâncias. Esta limitação impede a exploração da complementariedade dos modelos de forma ótima. Ao remover um modelo de um *ensemble*, remove-se todos os seus acertos, o que produz dois efeitos prejudiciais: (i) perda de voto (influência) em direção a classe correta e (ii) o *ensemble* perde a capacidade de acertar a predição de instâncias onde somente o modelo removido acertou. Para contornar esta limitação diferentes trabalhos apresentam técnicas de seleção dinâmica de *ensembles Dynamic Ensemble Selection* (DES) [47, 42, 41, 35, 21]. O objetivo da seleção dinâmica é permitir que o voto de um modelo base seja anulado (removido) somente quando conveniente (e.g., quando o modelo errou a predição). Nesta linha Nguyen et al. [47] apresentam uma abordagem para seleção de modelos em *ensembles* com base na confiança dos modelos. Para cada modelo os autores estabelecem um limiar de credibilidade com base na minimização da perda empírica 0-1 no conjunto de treino. Durante a fase de teste, para cada instância os autores computam a entropia na predição de cada modelo como reflexo de sua confiança. Caso um modelo qualquer tenha confiança inferior ao seu limiar de credibilidade para uma instância específica, sua predição é descartada. Os autores comparam a abordagem apresentada com outras *baselines* da literatura sobre 62 base de dados, apresentando resultados superiores aos demais *baselines* em diferentes domínios. Trabalhos recentes apresentam estratégias para seleção dinâmica de modelos em *ensembles* através do uso de modelos de atenção conhecidas como *adaptive ensemble learning* (Liu et al. [41, 42]). Tais abordagens combinam a representação das instâncias com as predições dos modelos do *ensemble* em mecanismos de atenção. Durante o treino o módulo de atenção aprende quando ofuscar e ampliar a participação dos modelos no *ensemble* correlacionando a distribuição de importância dos modelos no *ensemble*, as representações das instâncias e o desempenho do *ensemble* (acertos e falhas nas predições). Para guiar o treinamento dos modelos tais estratégias contam com uma função de perda que visa maximizar a acurácia dos modelos e também a diversidade evitando *overfitting*. Desta forma, durante a fase do modelo via *backpropa-*

gation este aprende a ajustar os mecanismos de atenção para gerar pesos que otimizem a participação dos modelos do ensemble de acordo com cada instância. Tais estratégias apresentam resultados promissores, porém são significativamente sensíveis a parametrização, sendo necessário profundo conhecimento para o ajuste adequado de parâmetros.

Similar as estratégias de *adaptive ensemble learning*, estratégias DES também focam na estimação da performance dos modelos do *ensemble*, porém com base em zonas de competência [21, 35]. Cruz et al. [21] estendem o *framework* proposto em Oliveira et al. [49] introduzindo a *framework* FIRE-DES++. O foco do FIRE-DES++ é reduzir o ensemble para um sub conjunto de modelos confiáveis e diversificados. Para isso os autores utilizam o conceito de regiões de competência. Dada uma instância presente no conjunto de teste, uma região de competência consiste de instâncias próximas (*k-nearest neighbors*) a instância de teste retiradas de um conjunto de validação a parte. Estas regiões são utilizadas para medir o desempenho de cada modelo base sobre cada instância e também a diversidade do *ensemble*. Dado o foco dos autores em problemas de classificação altamente desbalanceados (e.g., análise de risco crédito) os autores aperfeiçoam a região de competência para medir o desempenho dos modelos de *ensemble* sem a influência de ruídos, *outliers* e do próprio desbalanceamento. Desta forma, os autores selecionam os classificadores com melhor desempenho e contribuição para diversificação e realizam um *majority voting* dos modelos finais. Nesta linha Hou et al. [35] introduzem uma nova estratégia denominada META-DESKNN-MI. O foco da estratégia também é na seleção de *ensembles* avaliados em regiões de confiança de vizinhos próximos as instâncias. No entanto para seleção dos *ensembles* os autores empregam um outro modelo de classificação para aprender a selecionar modelos denominado META-DES. Para treinar o META-DES os autores introduzem um novo conjunto de MFs voltadas para o desempenho e diversidade dos modelos do *ensemble*. Após aplicar o META-DES para seleção de ensembles os autores utilizam *majority voting* para predição das instâncias. Ambos trabalhos [21, 35] apresentam grande robustez na seleção de *ensembles* com foco em problemas de elevado desbalanceamento entre classes. No entanto, tais técnicas são extremamente suscetíveis a qualidade das regiões de competência de modo que o posicionamento (e.g., região de borda mal definida) da instância no espaço de atributos pode impedir a construção de boas regiões de competência.

Similar aos trabalhos voltados par DES previamente abordados, nesta dissertação focamos nossos esforços na seleção de modelos base para *stacking* a nível de instância. No entanto, diferente dos trabalhos que utilizam regiões de competência neste trabalho utilizamos estimadores de desempenho treinados em um novo conjunto de MFs criados sobre o conjunto de dados de treino que contem informações coletivas e individual dos modelos base, além de atributos estatísticos da base de dados.

Capítulo 4

Metodologia

Neste capítulo descrevemos a nossa metodologia e o *framework* PTFS (Previsão e Tratamento de Falhas em modelos de *Stacking*). Na seção apresentamos conceitos fundamentais para a metodologia deste trabalho e as Questões de Pesquisa que guiam este estudo. Na seção 4.2 detalhamos o nosso *framework* e as três estratégias propostas para aumentar a efetividade do meta-classificador do *stacking* com foco no tratamento de falhas de predição dos modelos base do *stacking*.

4.1 Tratamento de Falhas

A efetividade de um modelo de *stacking* depende da efetividade dos modelos base que compõe o *stacking* [41, 21, 35]. Meta-classificadores tradicionais são prejudicados pelas falhas de predição dos modelos base. *Falha de Predição* de um modelo consiste no ato do classificador atribuir uma classe para um documento diferente da sua classe verdadeira. Por exemplo, quando um modelo atribui a um documento a classe esporte quando na verdade este é um documento pertencente a classe política, o modelo comete uma Falha de Predição. O meta-classificador se alimenta das probabilidades (predições) dos modelos base. Quando um modelo base comete uma Falha de Predição este modelo envia para o meta-classificador informação errada, que por vez influencia o meta-classificador a também falhar na predição. No cenário ideal o meta-classificador deve ser robusto o suficiente para desconsiderar modelos que cometeram falhas de predição, preservando os demais. Denominamos este cenário como *Upperbound*. Em cada investigação realizada neste trabalho, buscamos estudar formas de aumentar a efetividade do *stacking* prevendo e tratando Falhas de Predição. Deste modo, cada uma das estratégias propostas para tratamento de Falhas de Predição dos modelos base descritas nas próxima seção possuem um *Upperbound* específico e particular que revela o potencial máximo da estratégia. Nesta linha, nosso objetivo é propor e avaliar novas estratégias para aumentar a efetividade de modelos de *stacking* através da previsão e tratamento de falhas, buscando atingir o

Upperbound de tais estratégias. Mais especificamente, buscamos responder as seguintes Questões de Pesquisa (QPs):

- *QP1 - É possível aproximar a efetividade do* Upperbound *do* stacking *através da previsão de falha dos modelos base?*
- *QP2 - É possível aproximar a efetividade do* Upperbound *do* stacking *identificando o melhor modelo base para cada documento?*
- *QP3 - É possível treinar meta-classificadores capazes de predizer corretamente a classe de documentos onde a maioria dos modelos base falham?*

Com QP1 buscamos verificar se é possível atingir o *Upperbound* do *stacking* prevendo e anulando Falhas de Predição dos modelos base. Anular a Falha de Predição de um modelo base consiste no ato de atribuir zero a distribuição de probabilidades estimada pelo modelo base para um documento, dado que o modelo cometeu uma Falha de Predição. Através do *Upperbound* simulamos um cenário onde todas as Falhas de Predição são anuladas mostrando o potencial do meta-classificador sem o prejuízo causado pelas falhas dos modelos base.

QP2 consiste em uma alteração do problema originalmente estabelecido em QP1. Para atingir o *Upperbound* de QP1 precisamos identificar em meio a K modelos base quais falharam. Neste contexto, existem dois possíveis casos de insucesso na previsão de falhas: prever que um modelo base acertou a predição quando este falhou (falso negativo) e prever que o modelo base falhou quando este acertou (falso positivo). Tomando como base o *stacking* tradicional, o meta-classificador toma decisão com base nas predições de todos os modelos base (independente se cometeram Falha de Predição ou não). Supondo que o estimador de falhas preveja que todos modelos base sempre acertam a predição (mesmo quando os modelos base erram atingindo o total de falso positivos) por consequente mantendo a participação dos modelos base, neste cenário o *ensemble* formando consiste no *stacking* tradicional. Portanto, prever falso negativos implica em pouco ou nenhum prejuízo na efetividade pré estabelecida pelo *stacking* tradicional. No entanto, supondo que o estimador de falhas tenha identificado equivocadamente que um modelo base cometeu uma Falha de Predição, esta falha degrada a efetividade pré estabelecida pelo *stacking* tradicional, pois remove-se predições corretas prejudicando a tomada de decisão do meta-classificador. Portanto, anular predições de modelos base sem prejudicar a efetividade do *stacking* devido a falso positivos é uma tarefa desafiadora, principalmente em *stackings* com muitos modelos. Nesta linha, em QP2 focamos em encontrar para cada documento o modelo mais adequado (que acertou a predição com mais confiança) em vez de prever quais falham. Neste cenário, mesmo ao selecionar um modelo que não seja o mais adequado, selecionar um modelo que sucedeu porém com menos confiança implica no mesmo resultado prático que escolher o modelo mais adequado em termos de efetividade.

O *Upperbound* desta investigação consiste no cenário onde para cada documento em que ao menos um modelo base acertou a predição, um modelo assertivo na predição deve ser escolhido.

Finalmente, em QP3 verificamos se é possível treinar meta-classificadores mais robustos contra falhas de modelos base. Mais especificamente, queremos verificar se é possível aumentar a efetividade do *stacking* tratando *especificamente e de forma diferenciada* casos de falha do meta-classificador em documentos onde a maioria dos modelos base falham. Estes documentos são desafiadores para os meta-classificadores e também para o tratamento de falhas em QP1 (necessário identificar maior número de modelos que falham na predição) e também para QP2 (escolher um modelo correto em meio a maioria errada). Denominamos estes documentos como: **documentos difíceis** (e os demais como **documentos fáceis**). Nosso objetivo é estender as estratégias de previsão de falhas investigadas nas questões prévias para destinar documentos difíceis para um classificar especialista nestes. Diferente das investigações em QP1 e QP2 onde o fator determinante para a o sucesso na aproximação do *Upperbound* recai sobre a previsão de falhas antes de chegar ao meta-classificador, neste cenário buscamos treinar um segundo meta-classificador especializado em classificar documentos difíceis, denominado meta-classificador **Especialista**. Este meta-classificador deve realizar a predição de documentos difíceis enquanto o meta-classificador original, denominado **Tradicional** deve realizar a predição dos demais. Desta forma, definimos o *Upperbound* desta questão de pesquisa como a separação ideal de todos os documentos difíceis, onde a previsão de cada documento é realizada pelo meta-classificador específico para sua condição (fácil ou difícil).

Na próxima seção descrevemos o *framework* PTFS e as três estratégias de melhoria de efetividade do *stacking* contidas em nosso *framework*.

4.2 Framework

Nesta seção descrevemos as estratégias contidas no *framework* PTFS. Na seção 4.2.1 descrevemos a estratégia Error Detecion, proposta para responder QP1. Na seção 4.2.2 descrevemos a estratégia Best Model, proposta para responder QP2. Por fim, na seção 4.2.3 descrevemos a estratégia Har Docs Detecion, proposta para responder QP3.

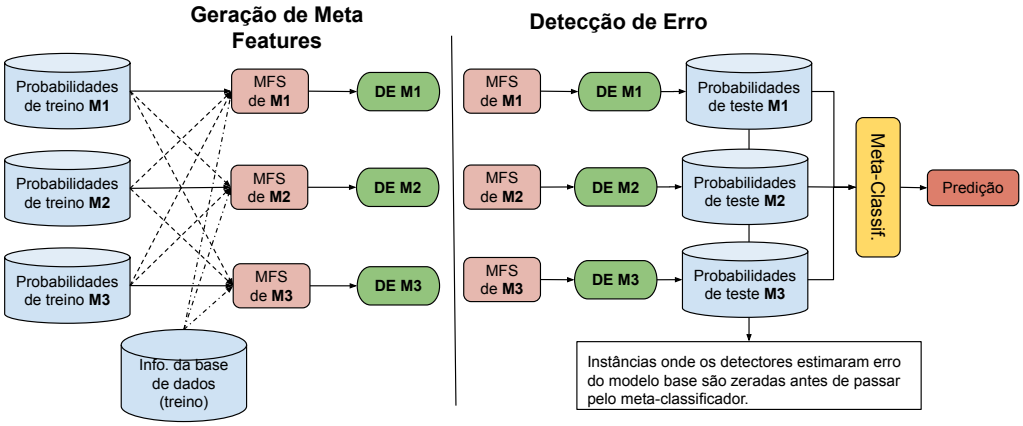

Figura 4.1: Visão geral das etapas de treino e predição da estratégia Error Detection. As MFs utilizadas no teste são geradas da mesma forma que as produzidas no treino.

4.2.1 Error Detection

O desempenho de um meta-classificador é dependente da efetividade dos modelos base do *stacking*. Dado a diversidade que beneficia os estratégias de *ensemble* sabe-se que para diferentes instâncias de uma base de dados (e.g., documentos) existem modelos mais adequados para classificação destes [41, 21, 35]. Tratando-se do *stacking* tradicional, no cenário ideal, o meta-classificador deve em meio a um conjunto de classificadores considerar modelos base assertivos na predição da classe de um documento e desconsiderar os demais. Ao estender este comportamento para todos documentos da base atinge-se o que definimos como *Upperbound* para a estratégia Error Detection. O foco desta estratégia consiste em prever quais modelos falham na predição de cada instância no conjunto de dados de teste e "anular" a predição dos modelos falhos, aplicando zero em suas distribuições de probabilidade. Prever falhas dos modelos base permite que isolemos Falhas de Predição impedindo que estas prejudiquem o desempenho do meta-classificador deste modo atingindo o *Upperbound*, objetivo da primeira Questão de Pesquisa deste trapalho:

QP1 - E possível aproximar a efetividade do limite superior do stacking *a partir da previsão da falha dos modelos base?*

A estratégia Error Detection adiciona uma etapa no processo tradicional de *stacking* que consiste na previsão e anulação de Falhas de Predição antes de passar as predições dos modelos base para o meta-classificador. Para detectar as falhas dos modelos do *stacking*, treinamos para cada modelo base, um segundo modelo para prever se o modelo base errou. Formalmente, para cada modelo base $m_k \in M$ do *stacking* treinamos um Estimador de Falhas (EF) $e_{mk} \in E$ para aprender a prever as falhas de m_k. Para treinar e_{mk}, primeiro criamos para cada documento $d_i \in D_{train}$ um novo rótulo y_{d_i} que representa se o modelo base m_k acertou a predição de $d_i (y_{d_i} = 1)$ ou errou $(y_{d_i} = 0)$. Para verificar se m_k

errou a predição utilizamos as suas probabilidades geradas para o treino da meta-camada (vide seção 5.3). Como representação para o documento d_i utilizamos um conjunto de MFs descritas na seção 5.5. Na figura 4.1 na etapa de geração de MFs podemos ver o compartilhamento entre as probabilidades dos modelos e informação da base de dados no processo de geração de MFs para cada detector de erro de cada classificador base.

Durante a fase de predição (teste), para cada modelo base m_k estimamos se a predição $m_k(d_i)$ (predição do modelo m_k sobre o documento $d_i \in D_{test}$) consiste em uma Falha de Predição. Caso sim, anulamos sua predição atribuindo zeros a distribuição de probabilidades $m_k(d_i)$. Ao anular a predição de um modelo base para um documento, removemos a sua contribuição na tomada de decisão do meta-classificador. Na Figura 4.1, na fase de detecção ilustramos a intermediação de cada detector de erro antes das probabilidades chegar no meta-classificador.

O maior desafio da estratégia Error Detection consiste em evitar falso positivos (predizer que um modelo falhou quando este acertou a predição), pois neste case estamos removendo informação benéfica do *stacking*, causando regressão na efetividade inicial do meta-classificador.

4.2.2 Best Model

A partir da questão QP1 podemos verificar se é possível encontrar sub-conjuntos de modelos base adequados para diferentes documentos por meio da estratégia Error Detection. Apesar de encontrar bons sub conjuntos de modelos ótimos ser uma forma de aumentar a probabilidade de sucesso do meta-classificador, esta forma de abordar o problema possui a desvantagem de ser demasiadamente sensível aos casos de falso positivos na previsão de falhas, como descrito na seção 4.1. Supondo que para um documento d_i qualquer, se múltiplos estimadores de erro preverem falhas de modelos base equivocadamente, predições importantes para o meta-classificador são anuladas e consequentemente aumenta-se consideravelmente as chances do meta-classificador falhar. Uma forma de contornar esta limitação consiste na simplificação do problema alterando o objetivo de identificar quais modelos base falharam para identificar qual é o modelo base mais adequado para a predição de um documento específico. Analisar a viabilidade de encontrar o melhor modelo em um *stacking* para cada documento é o foco da segunda Questão de Pesquisa deste trabalho:

QP2 - É possível aproximar o upperbound *do* stacking *identificando o melhor modelo base para cada documento?*

Para responder QP2 apresentamos a segunda estratégia do nosso *framework* Best

Model. Nesta estratégia o nosso objetivo é encontrar o modelo mais adequado para predição da classe de um documento específico, em meio ao conjunto de modelos base. Definimos como o modelo mais adequado para predição da classe de um documento aquele que acertar a predição com maior confiança. Nos casos onde todos os modelos base falham, o mais adequado é o modelo que falhou com menor confiança. Para esta estratégia o *Upperbound* consiste em escolher para todo documento o modelo mais adequado (acertou a predição com maior confiança). Vale ressaltar que, equivalente a encontrar o *Upperbound* para os documentos onde mais de um modelo base sucede na predição, escolher qualquer um dos que sucederam resulta na mesma efetividade, característica que aumenta a tolerância a Falhas de Predição da estratégia sem prejudicar a efetividade geral. Na Figura 4.2 apresentamos a visão geral da estratégia para identificar o melhor modelo para predição de um documento específico em um *ensemble*.

Formalmente, para identificar o modelo mais adequado ($Best(M)$) para realizar a predição de cada documento treinamos um novo classificador denominado *Best Model Detector* (BMD). Para treinar o BMD transformamos os dados para atender o formato do problema. Primeiro criamos novos rótulos $y_{d_i} \in Y$ para cada documento $d_i \in D_{train}$. Cada rótulo Y_{d_i} é uma referência para o modelo base $Best(M)$ que acertou a predição de d_i com maior confiança. Para representar os documentos utilizamos as MFs descritas na seção 5.5.

Na fase de predição para cada documento $d_i \in D_{test}$ utilizamos o modelo BMD para identificar o modelo $Best(M)$ para d_i. Identificado o melhor modelo usamos a predição deste como a predição final para o documento. Importante destacar que, apesar de cada rótulo no treino do BMD conter o classificador que acertou com maior confiança, basta o BMD escolher um modelo correto que a efetividade será equivalente a escolha do modelo mais adequado.

Apesar de amenizar a limitação de identificar o conjunto de modelos mais apropriado para o modelo mais adequado, a estratégia Best Model é sensível a documentos

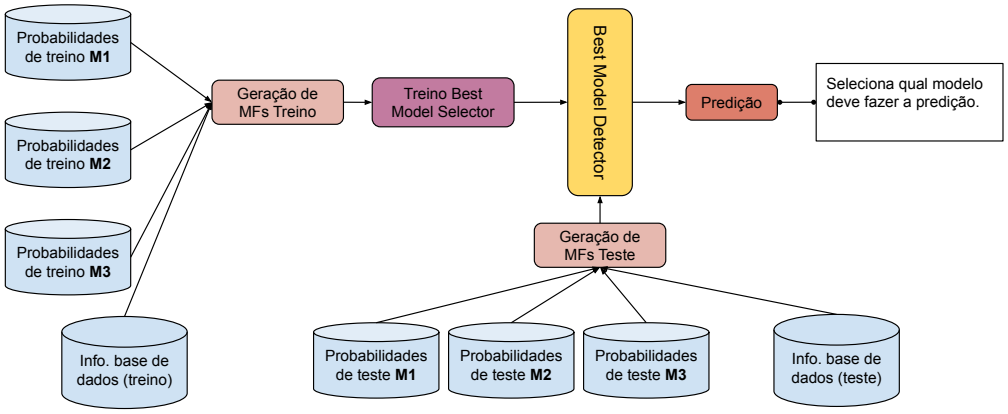

Figura 4.2: Visão geral da estratégia *Best Model*.

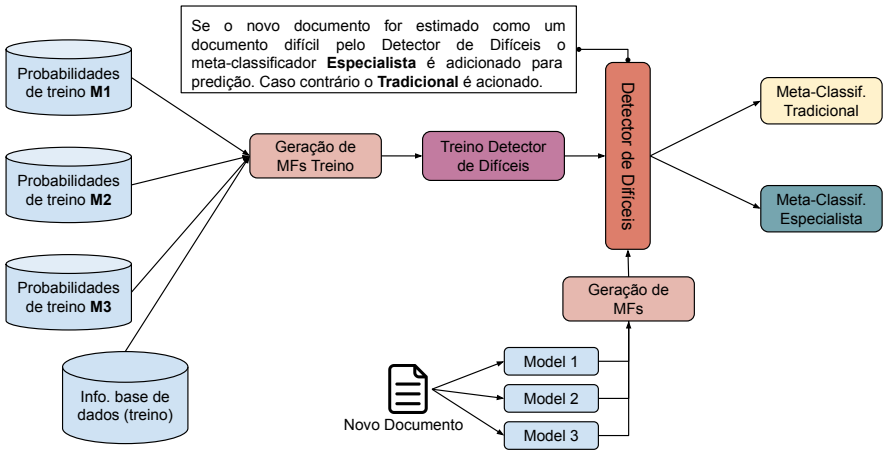

Figura 4.3: Visão geral da estratégia Hard Docs Detection

onde maioria dos modelos base falham. Na próxima seção, abordamos QP3, voltada para o problema da falha de múltiplos modelos base e a terceira estratégia apresentada neste trabalho utilizada para tratar QP3.

4.2.3 Hard Docs Detection

Como descrito na seção 4.2.2 a estratégia Best Model, assim como o *stacking* tradicional é mais propenso a falhar em documentos onde poucos modelos base acertam. Observamos que as falhas do meta-classificador concentram-se majoritariamente sobre documentos em que mais de 50% dos modelos base falharam. Denominamos estes documentos como **Documentos Difíceis** e os demais como **Documentos Fáceis**. Este comportamento prejudica o meta-classificador ao enviesá-lo em favor dos modelos base que falham na predição, consequentemente ofuscando os modelos que sucedem na predição. Neste linha, pretendemos responder a terceira Questão de Pesquisa deste trabalho:

QP3 - É possível treinar meta-classificadores capazes de predizer corretamente a classe de documentos onde a maioria dos modelos base falham?

Supondo que seja viável treinar um meta-classificador mais apto a lidar com documentos onde maioria dos modelos base falham, a conversão destes casos de falha para sucesso trataria considerável parte dos problemas do meta-classificador. Para investigar esta hipótese precisa-se tratar dois desafios: identificar documentos difíceis e criar um meta-classificador mais robusto contra documentos difíceis. Para tratar QP3, introduzimos a terceira estratégia do nosso *framework* Hard Docs Detection. Esta estratégia consiste em um pipeline de duas etapas voltado para o tratamento isolado de documentos

difíceis: identificação de documentos difíceis e classificação de documentos.

Na etapa de identificação de documentos difíceis nós treinamos um classificador denominado **Detector de documentos Difíceis** (DD) para identificar os documentos mais difíceis para o classificador tradicional. O treino do detector de difíceis se faz como se segue: para cada documento d_i em D_{train} verificamos quantos modelos base acertaram a predição deste documento. Para isso utilizamos as predições dos modelos base sobre $(m_k(d_i), \quad \forall \quad m_k(di) \in M)$. Para documentos onde a maioria dos modelos base ($> 50\%$) falharam na predição criamos um novo rótulo ($y_{d_i} = 1$) indicando este documento como difícil. Os demais documentos recebem rótulo de fáceis ($y_{d_i} = 0$). Como estratégia de representação dos documentos utilizamos a mesma representação com MFs (vide seção 5.5) utilizada nas duas abordagens apresentadas previamente. Na Figura 4.3 apresenta-se a visualização completa da estratégia, deste o treino do Detector de Difíceis e até a predição de um novo documento.

Na segunda etapa do pipeline (classificação de documentos) treinamos dois meta-classificadores: **Especialista** e **Tradicional**. O meta-classificador especialista, como próprio nome diz é especialista em tratar documentos difíceis. Documentos difíceis geralmente correspondem a pequenas parcelas das bases de dados. Apontamos esta comportamento como um dos principais fatores que contribuem para a falha do meta-classificador. Portanto, para tratar este problema utilizamos a técnica *Oversampling* para equilibrar o volume de documentos difíceis e fáceis no treino do meta-classificador (D_{train}). No aumento de dados replicamos a quantidade de documentos difíceis até o volume de difíceis se equiparar ao volume de fáceis. Com o novo conjunto de treino aumentado espera-se aumentar a robustez do meta-classificador especialista contra documentos difíceis. O meta-classificador tradicional é treinado com o conjunto de dados original seguindo os procedimentos usuais.

Durante a etapa de predição, para cada documento $d_i \in D_{test}$ primeiro passamos este documento pelo DD. Caso o DD classificar este documento como difícil encaminhamos d_i para o meta-classificador especialista. Caso contrário, para o meta-classificador tradicional. Importante notar que para estimar o limite superior desta estratégia simulamos o DD ideal (detector que sempre sabe se um documento é difícil o fácil).

Devido a característica de possuir duas etapas que dependem de modelos de aprendizado, esta estratégia possui dois focos de melhoria: identificar documentos difíceis e a classificação correta destes. Simulando um identificador de documentos difíceis ideal, basta um meta-classificador ligeiramente melhor que o tradicional no tratamento de documentos difíceis, para que melhorias na efetividade do *stacking* ocorram. Desta forma, estabelecemos como *Upperbound* da estratégia o separador de documentos difíceis ideal, que identifica completa e corretamente todos os documentos assim definidos.

Capítulo 5

Ambiente Experimental

Neste capítulo apresentamos as configurações do ambiente utilizado para executar os experimentos apresentados neste estudo. Na seção 5.1 apresentamos as bases de dados utilizadas para avaliação do *stacking* e das estratégias do nosso *framework*. Na seção 5.2 descrevemos quais algoritmos e representações utilizamos para construir os modelos de *stacking* base deste estudo, assim como as suas parametrizações. Na seção 5.3 descrevemos o processo de treino e predição do *stacking*. Na seção 5.4 apresentamos as métricas utilizadas para medir a efetividade dos modelos de *stacking* e as estratégias do nosso *framework*. Apresentamos também a infraestrutura utilizada para execução dos experimentos. Na seção 5.5 introduzimos as *meta-features* utilizadas pelos modelos de tratamento e previsão de falhas nas estratégias do nosso *framework*. Por fim, na seção 5.6 descrevemos o processo de calibragem dos modelos base do *stacking*.

5.1 Base de Dados

Para este estudo consideramos seis bases de dados amplamente utilizadas como *benchmark* em CAD na literatura. Dessas seis bases de dados, três são bases de dados de médio-grande porte (mais de 100.000 documentos) [25, 66] – *AG's News* (AGNews), Avaliações do *Internet Movie Dataset* (IMDb) e *Sogou News Dataset* (Sogou) – e três são bases de dados de tamanho pequeno-médio [14, 16] – *20 Newsgroup* (20NG), *ACM Digital Library* (ACM) e *Wolrd Wide Knowledge Base* (WebKB).

Na Tabela 5.1 temos um resumo de cada uma das bases de dados. Apresentamos o número de documentos (#Docs), número de classes (#Classes) de cada base e informações sobre a distribuição de documentos entre as classes (Distribuição de Classes). Nas Distribuições de Classes, podemos ver o tamanho da classe minoritária (Menor), quantidade mediana (Mediana), quantidade média (Média) e a tamanho da classe majoritária (Maior). É importante destacar todas essas informações, já que as diferentes características dessas bases de dados levam a desafios diferentes para os algoritmos CAD.

Abaixo temos as informações dos dados textuais e das classes para cada base de dados:

Tabela 5.1: Bases de Dados

Bases de Dados	#Doc.	#Classes	Distribuição de Classes				Skewness
			Menor	Mediana	Média	Maior	
20NG	18846	20	628	984	942	999	Balanceada
ACM	24897	11	63	2041	2263	6565	Desbalanceada
AGNews	1276	4	31900	31900	31900	31900	Balanceada
IMDb	348415	10	12836	31551	34841	63233	Desbalanceada
Sogou	510000	5	102000	102000	102000	102000	Balanceada
WebKB	8199	7	137	926	1171	3705	Desbalanceada

- 20 Newsgroups (20NG): consiste em 18,846 documentos divididos em 20 categorias de notícias, sendo que cada categoria aborda um tópico específico (por exemplo, política, religião, esporte, etc.). É um conjunto de dados popular na área de CAD, pois possui a característica de ter documentos com quantidades aproximadamente iguais entre as diferentes 20 classes (ou seja, é um conjunto de dados balanceado).

- ACM Digital Library (ACM): composta por 24,897 documentos da Biblioteca Digital ACM. Todos os artigos são do campo da Ciência da Computação, e o objetivo é classificar cada artigo em 11 classes do primeiro nível da taxonomia adotada pela CAD.

- AG's News (AGNews): conjunto de artigos de notícias coletados da Web, contendo 127,600 documentos e 4 classes. Os documentos vêm de 496,835 artigos de notícias categorizados e mais de 2,000 fontes de notícias. Os rótulos foram criados de acordo com o título e a descrição de cada artigo. Este conjunto de dados possui um equilíbrio igual para todas as classes, cada uma tendo a mesma quantidade de documentos.

- Avaliações do *Internet Movie Database* (IMDb): conjunto criado a partir do site de avaliações de filmes IMDb, onde os criadores do conjunto de dados selecionaram 50,000 filmes aleatórios e coletaram todas as suas avaliações. Este conjunto de dados tem dez classes, que se referentes às avaliações dos usuários (escaladas de 0 a 10) nos filmes selecionados. O conjunto de dados resultante tem 348,415 documentos e um alto desequilíbrio de documentos nas dez classes.

- World Wide Knowledge Base (WebKB): Este conjunto de dados foi coletado dos departamentos de Ciência da Computação de várias universidades em janeiro de 1997 pelo projeto World Wide Knowledge Base (WebKB) do grupo da Universidade Carnegie Mellon (CMU). Há um total de 8,199 páginas da web que foram

coletadas e que foram separadas nas seguintes categorias: estudante, corpo docente, funcionários, departamento, curso, projeto e outros.

- Notícias Sogou (Sogou): Criado a partir do mecanismo de busca chinês Sogou, o conjunto de dados foi gerado a partir de um total de 2,909,551 artigos de notícias de diferentes tópicos. O conjunto de dados resultante tem 510,000 documentos e cinco classes balanceadas (esportes, finanças, entretenimento, automóveis e tecnologia). Como é um conjunto de dados chinês, foi necessário fazer um pré-processamento específico para transformar o conjunto de dados em Pinyin – a romanização fonética do chinês. O pré-processamento foi feito com o pacote de sistema de segmentação chinesa *pypinyin* com *jieba*, conforme indicado pelos autores em [66].

5.2 Representações e Algoritmos Supervisionados

Como base para os modelos de *stacking* utilizados neste trabalho, buscamos inspiração no trabalho de Gomes et al [22]. Naquele trabalho os autores avaliaram *stackings* de 18 modelos. Dos 18 modelos 16 são combinações entre as representações TF-IDF, PTE [60], FastText [9] e Similarity-Basesd-Meta-Features [15] com os algoritmos LinearSVM [29], Regressão Logística [29], XGBoost [19] e kNN [3]. Além disso, contam com dois modelos profundos – BERT [24] e XLNet –[64]. Dentre os 18 modelos contidos no trabalho escolhemos para nossos estudos os sete seguintes modelos descritos na tabela 5.2:

Tabela 5.2: Modelos utilizados com base do *stacking*. Os demais parâmetros não especificados para os modelos profundos foram configurados como padrão do *HuggingFace* [34]

Sigla	Modelos	Representação	Parâmetros	Valor
SVM-TFIDF	SVM	TF-IDF		
SVM-MF	SVM	SM-Meta-Features		
LR-TFIDF	Logistic Reg.	TF-IDF	Padrão Scikit-Learn	
kNN-TFIDF	kNN	TF-IDF		
kNN-MF	kNN	SM-Meta-Features		
BERT	BERT	-	Pretrained Model	Base
XLNet	XLNet	-	batch_size	64
			learning_rate	5e-5
			max_lenght	128

Com este grupo de modelos conseguimos obter *stackings* eficientes (menor consumo computacional) e com efetividade próxima aos estudos presentes em Gomes et al [22]. Para o meta-classificador mantivemos o algoritmo Regressão Logística. No treino do

meta-classificador, assim com realizado pelos autores, otimizamos os hiper parâmetros $C : [1 \times 10^{-8}, 20]$, $Penalty : \{None, l2\}$ e $ClassWeight : \{None, balanced\}$ com auxílio da biblioteca *Optuna* [2]. Como estimador de falhas na estratégia Error Detection e detector de documentos difíceis na estratégia Hard Docs utilizamos algoritmo *Random Forest* com os parâmetros padrão da biblioteca Scikit-Learn. Escolhemos este modelo devido ao seu baixo custo computacional, velocidade e elevado nível de efetividade obtido em nossos experimentos.

5.3 Stacking

Para aplicar a técnica de *stacking* é necessário construir representações em forma de distribuição de probabilidades para cada documento no treino e no teste, para cada modelo base do *stacking*. Desta forma, ao final do processo concatena-se para cada documento as probabilidades de cada modelo base sobre o documento, construindo sua representação final para utilização no meta-classificador. A geração de probabilidades para o meta-classificador é produzida em duas etapas: treino e teste.

Na construção das probabilidades do treino do meta-classificador deve-se subdividir o treino em *folds* em um processo de validação cruzada estratificada. Neste trabalho utilizamos quatro *folds*. Na Figura 5.1 podemos ver a divisão dos documentos de treino em quatro *folds*, onde utiliza-se três para treinar os modelos base do *stacking* e realiza-se a predição no *fold* restante. Repete-se este processo até todos os *folds* receberem predições de todos os modelos base. Ao final do processo concatena-se a probabilidade gerada sobre os *folds* de validação e assim constituí-se as representações de probabilidade utilizadas para o treino do meta-classificador. Com estas probabilidades o meta-classificador aprende a combinar as predições dos modelos com objetivo de maximizar a efetividade na tarefa de predição. Gerar as probabilidades do treino da meta-camada utilizando o processo de validação cruzada estratificada é importante para evitar a ocorrência de *overfitting* do meta-classificador [63].

Na etapa de teste a construção de representações de distribuição de probabilidades segue o modo usual do treinamento de modelos de aprendizado supervisionado. Como mostrado na Figura 5.2 utilizamos o conjunto de treino para treinar os modelos base do *stacking* e depois aplicamos os modelos treinados nos documentos de teste extraindo a distribuição de probabilidade de cada modelo base para cada documento no teste. No fim do processo concatena-se para cada documento as distribuições de probabilidade de cada modelo base e constrói-se a representação dos documentos do teste, posteriormente utilizados pela meta-camada para predição.

5.4 Métricas e Avaliação

Os experimentos nas bases de dados menores foram executados utilizando validação cruzada com 10 *folds*. Para as bases de dados grandes utilizamos validação cruzada com cinco *folds*. Aplicamos uma etapa de otimização de parâmetros da meta-camada utilizando *Bayesian Optimization* [6] com 30 iterações e validação cruzada interna com cinco *folds* para seleção dos parâmetros. Para avaliar o desempenho dos modelos de *stacking* e das estratégias concorrentes ao *stacking* utilizamos métrica MacroF1 [58]. Além da MacroF1, utilizamos precisão e revocação [8] para avaliar os modelos de previsão de erro, seleção do melhor classificador e detecção de documentos difíceis para as estratégias Error Detection, Best Model e Hard Docs, respectivamente.

Para execução dos modelos do *stacking* utilizamos o ambiente de computação em nuvem *Elastic Computing* (i.e., AWS EC2) da *Amazon Web Service*. Para executar os algoritmos de aprendizado de máquina tradicionais utilizamos instâncias da classe *c6a.8xlarge* que possuem 32 CPUs e 64GB de memória RAM. Para executar os modelos profundos utilizamos instâncias da classe *g5.2xlarge* que possuem 8 CPUs, 24GB de memória *RAM* e um processador gráfico NVIDIA A10G com 24GB de *VRAM*.

5.5 Meta-Features

Para prever falhas nos modelos base analisamos, para cada modelo, o seu comportamento individual e também em relação aos demais modelos para cada documento. Utilizamos também informações do comportamento dos modelos em relação a base de

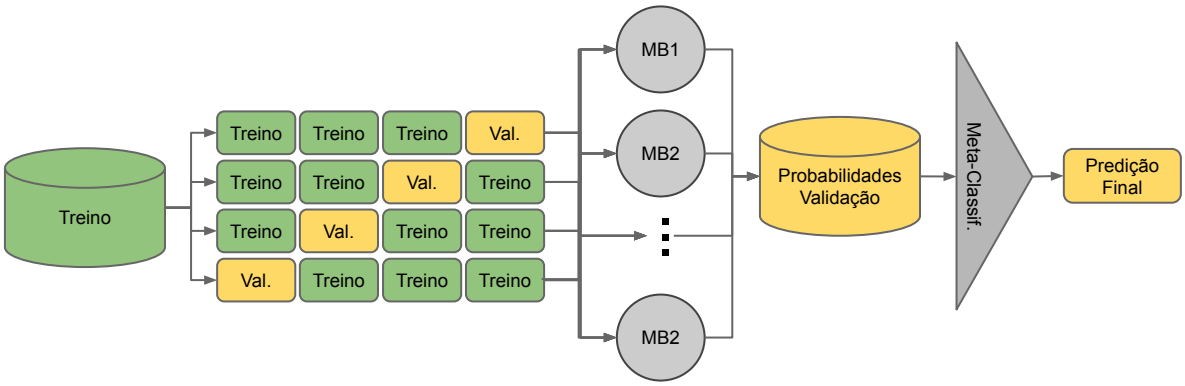

Figura 5.1: *Pipeline* de treinamento do meta-classificador.

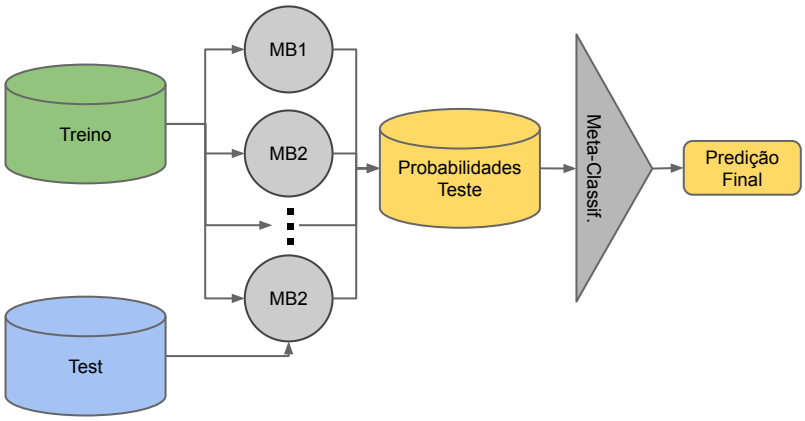

Figura 5.2: *Pipeline* de predição (teste) do meta-classificador.

dados. Obtemos essas informações de cada classificador a partir de um conjunto de sete novas MFs:

Tamanho da maior concordância: o tamanho da maior quantidade de modelos base que atribuiram a mesma classe para um determinado documento. Supondo que em um conjunto de sete modelos, se cinco modelos apontam para uma classe específica (e.g., cinco modelo classificam uma notícia como pertencente a categoria esporte), definimos o tamanho da maior concordância como 5.

Divergência: atributo binário que indica se o modelo base concorda com a maior concordância (se o modelo base faz parte da maior concordância). Supondo que em um conjunto de sete modelos, cinco escolhem uma classe específica, estes formam a maior concordância. Este atributo verifica se o modelo está entre os cinco contidos na maior concordância ou não.

Número de classes: tamanho do conjunto de classes únicas preditas pelos modelos base para um documento. Supondo que um documento (notícia) é classificado como esporte por três modelos, saúde por dois e tecnologia por um, o se número de classes únicas é três.

Entropia: entropia da distribuição de classes preditas para um documento. Verifica se os modelos estão em concordância computando a entropia sobre as predições do conjunto de modelos. Se em um conjunto de sete modelos, seis apontam para mesma classe e apenas um aponta para uma classe diferente, temos um baixo nível de entropia. Se sete modelos apontam para sete classes distintas entre si, temos o nível máximo de entropia.

Confiança: a probabilidade que o modelo atribuí a classe ao rotular o documento.

Taxa de acerto da classe: ao assinalar uma classe para um documento, computamos a razão entre o número de predições corretas pelo total de vezes em que o modelo base assinalou esta classe. Por exemplo, se um modelo classifica um documento como um novo documento como da categoria esporte, contabilizamos nos dados de validação (treino do meta-classificador), quantas vezes o modelo classificou sucedeu classificando

um documento como pertencente a categoria esporte.

Peso da classe: razão entre a quantidade de documentos da classe predita pelo modelo base pelo total de documentos da base. Se nos dados de validação o modelo classifica 100 documentos como esporte em um total de 1000 documentos, o peso desta classe para todo documento novo em relação a este modelo é 0,1.

5.6 Calibragem

Calibrar modelos de aprendizado de máquina consiste na tarefa de alinhar a confiança de um modelo com a real expectativa de um evento ocorrer (e.g., a probabilidade de um documento classificado como pertencente a categoria esporte ser realmente da categoria esporte). Supondo que um classificador realize 100 predições, com 80% de confiança cada, dizemos que este modelo está calibrado se sua acurácia também for de 80% sobre as 100 predições realizadas. Se sua acurácia for superior a 80% dizemos que o modelo está sub confiante (i.e., está estimando eventos com probabilidade inferior a probabilidade real destes eventos ocorrerem). Caso a acurácia do modelo seja menor que 80%, dizemos que o modelo está super confiante (i.e., está estimando eventos com probabilidades maiores que a real probabilidade destes eventos ocorrerem). Frequentemente modelos de classificação subestimam ou superestimam a probabilidade de um evento (um objeto pertencer a uma classe) produzindo confianças distorcidas [52, 48], principalmente modelos profundos devido principalmente a utilização da função *softmax* [31, 23]. Desta forma, um modelo com elevada acurácia não é necessariamente um modelo calibrado. O inverso também é válido.

Uma das investigações deste trabalho consiste em verificar se a calibragem de modelos base contribuí para melhora no desempenho geral de meta-classificadores na tarefa de CAD. Meta-classificadores geralmente utilizam predições de probabilidades de modelos base como representação (atributos). Nesta linha consideramos a hipótese de que predições com distribuições de probabilidade mais calibradas podem ajudar meta-classificadores a ponderar a importância das predições de modelos com maior assertividade.

Para calibrar os modelos tradicionais (não neurais) utilizamos o método *Isotonic Regression* [48]. Esta estratégia assume que a real distribuição de probabilidade de um evento (predição) i é dada por $y_i = m(f_i) + \epsilon_i$, sendo m uma função monotonicamente crescente e f_i a predição de um modelo. Dado um conjunto de treino (f_i, y_i) o objetivo desta estratégia é encontrar m, tal que $\hat{m} = argmin_z \sum (y_i - z(f_i))^2$.

Particularmente, modelos de linguagem pré-treinados geralmente são super confi-

antes [23]. Este comportamento é provocado pela utilização de *softmax* que normaliza a saídas dos modelos achatando pequenos valores de probabilidades e aumentando grandes valores. Para calibrar os modelos BERT e XLNet utilizamos a estratégia conhecida como *Temperature Scaling*. O objetivo desta estratégia é encontrar um escalar T que pondere a saída do modelo (*logits*) ($\hat{y}_i = softmax(\frac{1}{T} f_i)$) para que esta gere probabilidades mais próximas da real distribuição. Quando um modelo está super confiante (realizando predições com confiança superior a probabilidade real do evento) o valor de T geralmente é maior que 1, reduzindo a magnitude dos *logits* do modelo. O valor de T pode ser obtido através de diferentes processos de otimização. Neste trabalho utilizamos *Maximum Likelihood Estimation* como proposto em [23].

Capítulo 6

Resultados – Discussão

Neste capítulo apresentamos os resultados obtidos com as estratégias contida em nosso *framework* PTFS. Na seção 6.1 apresentamos os resultados obtidos com a estratégia Error Detection. Na seção 6.2 apresentamos os resultados obtidos com a estratégia Best Model. Por fim, na seção 6.3 apresentamos os resultados obtidos pela estratégia Hard Docs.

6.1 Detecção de Erro em Modelos Base do Stacking

Nesta seção apresentamos os resultados obtidos com a estratégia Error Detection. Na seção 6.1.1 apresentamos o desempenho da estratégia na classificação de documentos comparando com os resultados obtidos com o *stacking* tradicional. Na seção 6.1.2 apresentamos um estudo sobre o desempenho da estratégia de previsão de falhas dos modelos base.

6.1.1 Comparação Error Detection e Stacking

Na tabela 6.1 são apresentados os resultados obtidos pelo *stacking* e por cada estratégia proposta neste trabalho, além dos seus respectivos limites superiores (*Upper-bound*). Observando os resultados obtidos pela estratégia Error Detection, conseguimos responder a primeira questão de pesquisa QP1 deste trabalho. Podemos ver que sobre as diferentes bases de dados, o limite superior da estratégia supera os resultados obtidos pelo *stacking*, sendo em metade das bases por grandes margens. Sem calibragem: 17,03% (73,92 → 86,51) ACM, 13,79% (82,92 → 94,36) WebKB e 72,69% (35,27 → 60,91) IMDb. Calibrado: 18,68% (73,92 → 87,73) ACM, 13,89% (82,9 → 94,44) WebKB e 67,25% (35,27 → 58,99) IMDb. **Porém, não encontramos evidências que indiquem que a es-**

Tabela 6.1: Desempenho geral (MacroF1) do *stacking* e da estratégia Error Detection. Avaliamos o desempenho das estratégias utilizando probabilidades calibradas (Calib.) e sem calibrar (Normal). Para facilitar a comparação, destacamos em cinza os resultados obtidos com o *stacking* tradicional. Referenciamos o limite superior da estratégia Error Detection adicionando o prefixo "Upp." ao seu nome. Destacamos com triângulos verdes e vermelhos invertidos os ganhos e perdas estatísticos comparando cada estratégia com o *stacking* tradicional, respectivamente. Empates estatísticos destacamos com um círculo amarelo. Utilizamos *teste t student* com 95% de confiança.

	20NG		ACM		WEBKB	
	Normal	Calib.	Normal	Calib.	Normal	Calib.
Stacking	92.24	92.24 ●	73.92	73.76 ●	82.92	83.1 ●
Upp. Error Detection	96.7 ▲	96.93 ▲	86.51 ▲	87.73 ▲	94.36 ▲	94.44 ▲
Error Detection	91.99 ●	91.79 ●	73.81 ●	73.33 ●	82.37 ●	81.86 ●
	AGNews		IMDb		Sogou	
	Normal	Calib.	Normal	Calib.	Normal	Calib.
Stacking	94.9	94.92 ●	35.27	35.22 ●	96.87	96.88 ●
Upp. Error Detection	97.59 ▲	97.73 ▲	60.91 ▲	58.99 ▲	98.64 ▲	98.77 ▲
Error Detection	94.68 ▼	94.73 ▼	17.64 ▼	25.16 ▼	96.41 ▼	96.45 ▼

tratégia Error Detection é capaz de superar os resultados do *stacking* com probabilidades não calibradas e calibradas. Desta forma, respondemos negativamente a primeira Questão de Pesquisa deste trabalho - QP1.

No entanto, a estratégia ainda sim apresenta resultados muito próximos ao *stacking* tradicional em parte das bases de dados, empatando nas bases 20NG, ACM e WebKB e com resultados ligeiramente inferiores nas bases AGNews e Sogou. A base de dados IMDb se apresentou como a base mais desafiadora para estratégia, com perdas de 50% (35,27 → 17,64) e 28,66% (35,27 → 25,16) em relação ao *stacking* original, sem e com calibragem, respectivamente.

A diferença entre os resultados obtidos em relação ao Upperbound mostram a importância da necessidade de evoluir a estratégia de detecção de falhas. Na próxima seção analisamos o desempenho dos modelos de detecção de falhas dos modelos base e buscamos identificar focos de melhoria para esta estratégia.

6.1.2 Efetividade na Detecção de Falhas

O desempenho da estrategia **Error Detection** é diretamente impactado pelo desempenho dos EFs. Um EF pode falhar de duas formas: manter predições de modelos base que falharam ou excluir predições de modelos base que acertaram a predição. Remover predições corretas é demasiadamente prejudicial, pois degrada o estado inicial do

Tabela 6.2: Desempenho (Precisão e *Recall*) dos EFs dos classificadores base do *stacking* nas versões não calibradas (Norm.) e calibrada (Calib.).

Base Models		ACM	20NG	WebKB	AGNews	IMDb	Sogou
BERT	P	68,55	79,05	60,45	65,44	72,51	75,13
Norm.	R	55,67	69,07	31,58	32,39	93,82	58,77
BERT	P	69,91	78,33	60,15	63,65	72,35	75,96
Calib.	R	56,19	68,28	35,07	34,58	92,43	57,74
KNN-MF	P	71,58	75,93	81,38	77,86	83,59	80,14
Norm.	R	56,57	55,82	64,34	55,77	97,5	52,59
KNN-MF	P	71,88	76,82	81,5	77,97	81,88	79,62
Calib.	R	55,78	56,79	63,98	55,33	94,66	51,99
KNN-TF	P	76,94	83,74	84,39	78,3	83,29	90,17
Norm.	R	70,47	78,54	71,4	61,48	97,16	79,03
KNN-TF	P	74,98	83,26	83,5	79,62	82,47	86,25
Calib.	R	64,83	77,51	71,05	59,83	95,28	72,08
SVM-MF	P	70,05	76,2	80,82	73,74	74,62	78,55
Norm.	R	51,68	47,86	62,28	45,82	93,62	51,12
SVM-MF	P	72,48	76,2	81,05	75,51	74,5	78,22
Calib.	R	53,64	55,9	63,02	46,18	93,36	48,25
SVM-TF	P	73,34	79,58	72,83	76,95	73,9	79,31
Norm.	R	57,31	58,39	52,45	46,94	94,28	53,56
SVM-TF	P	74,33	81,35	72,02	76,64	74,27	78,3
Calib.	R	57,01	60,23	52,93	46,59	92,94	53,66
LR-TF	P	76,07	78,83	83,99	74,82	71,75	82,29
Norm.	R	61,84	67,7	71,67	47,28	92,44	58,53
LR-TF	P	75,14	83,61	75,25	75,9	72,67	81,61
Calib.	R	59,38	71,74	58,83	47,46	91,81	59,17
XLNet	P	68,87	79,2	62,67	64,9	75,8	76,91
Norm.	R	57,86	71,47	40,47	32,68	92,97	64,78
XLNet	P	69,79	78,79	63,28	64,09	76,05	76,95
Calib.	R	57,42	71,69	39,79	33,53	91,64	64,96

stacking, diminuindo o número de predições corretas inicialmente existente nas representações. Deste modo, a precisão na detecção de falhas é de suma importância, seguida pelo *recall*.

Na tabela 6.2 apresentamos a precisão e *recall* obtidos pelos EFs para cada modelo base do *stacking* para as seis bases de dados utilizadas neste trabalho. No geral os EFs obtiveram considerável precisão sobre todos as bases de dados com média de 75.98%, sendo o mínimo 60,15% para o classificador BERT Calibrado na base WebKB e a máxima de 90,17% para o modelo KNN-TFIDF não calibrado na base sogou. Em comparação com a precisão os EFs apresentaram menor média de *recall* 67,81% com mínimo de 31,58% para o classificador BERT não calibrado sobre a base WebKB e o máximo de 97,5% para o modelo KNN-MF não calibrado sobre a base IMDb. Por fim os EFs obtiveram em média 67,81% de MacroF1 com mínimo de 41,09% para o modelo BERT não calibrado sobre

a base WebKB e o máximo de 90,01% para o modelo KNN-MF não calibrado na base IMDb. Pode-se perceber que a MacroF1 média foi impulsionada pela precisão e retraída pelo *recall*. Este comportamento é esperado dado que a precisão é mais importante para o desempenho da estratégia. No entanto, apenas em quatro situações a MacroF1 obtida pelos EFs ficou abaixo de 50 pontos e foram especificamente os modelos BERT e XLNET sobre as bases WebKB e AGNews, tanto em suas versões não calibradas quanto calibradas. Além disso, estes dois modelos são os que onde foi obtido menor MacroF1 média na previsão de falhas sobre as diferentes bases de dados, sendo o BERT com MacroF1 média de 61,72 calibrado e 61,18 não calibrado e o XLNet com 63,95 calibrado e 63,92 não calibrado.

Ambos BERT e XLNet são os modelos mais complexos do *stacking* e suspeitamos que a complexidade destes modelos pode ser um dos fatores que tornam a previsão de falhas destes mais desafiadora, dado que é um desafio presente na literatura obter explicabilidade das tomadas de decisões de tais modelos. Na contra mão, os EFs obtiveram maior desempenho na previsão de falhas dos modelos KNN-TFIDF calibrado e não calibrado, estratégias mais simples do conjunto de modelos base, corroborando com a existência da relação inversa entre complexidade do modelos e efetividade na previsão de falha.

Esses resultados, apesar de negativos, abrem novas possibilidades de pesquisa com foco na utilização de informação compartilhada entre modelos de um *ensembles* para previsão de falhas de modelos individuais. Nesta linha apontamos o desenvolvimento de novas MFs para previsão de falhas e novas estratégias de aprendizado profundo especializadas em previsão de falhas de modelos como frentes promissoras.

6.2 Seleção do Melhor Classificador em Stackings

Nesta seção apresentamos os resultados obtidos com a estratégia Best Model. Na seção 6.2.1 apresentamos o desempenho da estratégia na classificação de documentos comparando com os resultados obtidos com o *stacking* tradicional. Na seção 6.2.2 apresentamos um estudo sobre a efetividade do BMD na seleção do melhor modelo base e também analisando a tolerância da estratégia a falhas.

Tabela 6.3: Desempenho geral (MacroF1) da estratégia Best Model. Avaliamos o desempenho da estratégia utilizando probabilidades calibradas (Calib.) e sem calibrar (Normal). Para facilitar a comparação, destacamos em cinza os resultados obtidos com o *stacking* tradicional. Referenciamos o limite superior da estratégia adicionando o prefixo "Upp." ao seu nome. Destacamos com triângulos verdes e vermelhos invertidos os ganhos e perdas estatísticos comparando ao *stacking* tradicional. Empates estatísticos destacamos com um círculo amarelo. Utilizamos *teste t student* com 95% de confiança.

	20NG		ACM		WEBKB	
	Normal	Calib.	Normal	Calib.	Normal	Calib.
Stacking	92,24	92,24 ●	73,92	73,76 ●	82,92	83,1 ●
Upp. Best Model	96,6 ▲	96,67 ▲	85,96 ▲	86,65 ▲	94,15 ▲	93,87 ▲
Best Model	91,97 ●	92,09 ●	70,47 ▼	70,86 ▼	81,19 ●	80,45 ▼
	AGNews		IMDb		Sogou	
	Normal	Calib.	Normal	Calib.	Normal	Calib.
Stacking	94,9	94,92 ●	35,27	35,22 ●	96,87	96,88 ●
Upp. Best Model	98,04 ▲	98,06 ▲	66,12 ▲	61,08 ▲	98,84 ▲	98,83 ▲
Best Model	93,72 ▼	94,25 ▼	29,1 ▼	28,15 ▼	96,04 ▼	96,22 ▼

6.2.1 Comparação Best Model e Stacking

Similar a estratégia **Error Detection**, como mostrado na tabela 6.2.1, o limite superior da estratégia **Best Model** supera o *stacking* tradicional em todas as bases e com grandes margens percentuais em metade das bases de dados sem calibragem: 16,28% (73,92 → 85,96) ACM, 13,54% (82,92 → 94,15) WebKB e 76,12% (35,27 → 66,12) IMDb e calibrado: 17,22% (73,92 → 86,65) ACM, 13,2% (82,92 → 93,87) WebKB e 73,17% (35,27 → 61,08) IMDb.

Apesar das grandes margens de MacroF1 obtidas pelo *Upperbound* a não encontramos evidências que indiquem que a estratégia Best Model supera o *stacking* em nenhuma das bases de dados. Desta forma, os resultados apresentados pela estratégia Best Model respondem negativamente a segunda Questão de Pesquisa QP2 deste trabalho.

Apesar de não superar o *stacking*, a estratégia **Best Model** apresentou resultados ligeiramente inferiores a este, empatando estatisticamente com o *stacking* nas bases 20NG e na sua versão calibrada na base WebKB. Sobre as bases AGNews e Sogou a estratégia apresenta resultado ligeiramente inferiores (todos com menos de 1% de diferença do *stacking*). A base IMDb se mostrou desafiadora para a estrategia, sendo a base onde se encontram os piores resultados sendo ficando abaixo do *stacking* por 17,49% (35,27 → 29,21) e 20,18% (35,27 → 28,15) pontos percentuais sem calibrar e calibrado, respectivamente.

Na próxima seção analisamos a efetividade do modelo de seleção do melhor classi-

ficador do *stacking*.

6.2.2 Efetividade na Seleção do Melhor Modelo

Nesta seção investigamos o quão distante nossa atual implementação da estratégia Best Model está de atingir o *Upperbound*.

Como podemos ver na tabela 6.4 a identificação do modelo mais adequado para classificação é uma tarefa bastante desafiadora. A média de desempenho do BMD sobre as diferentes bases de dados foi de 31,61 pontos de MacroF1 sobre o *stacking* não calibrado e 28,94 sobre o *stacking* calibrado. Os melhores resultados foram obtidos sobre as bases AGNews e Sogou (sem calibrar) com MacroF1 de 33,65 e 36,86, respectivamente. Os piores resultados ocorreram nas bases IMDb com MacroF1 de 20,07 e 16,64 calibrado e sem calibrar, respectivamente. Este comportamento é um indicativo de que a estratégia é demasiadamente sensível a bases menores e desbalanceadas. Porém, é importante notar que, apesar de obter baixa efetividade na identificação do modelo mais adequado a estratégia obtém resultados similares ao do *stacking* na maioria dos cenários (exceto IMDb como mostrado na seção 6.2.1). Esta observação reforça a nossa hipótese de que: apesar do BMD não ter escolhido o modelo mais adequado, este ainda escolheu um modelo bem sucedido na predição, resultando no mesmo resultado prático.

Como descrito na seção 4.2.2, para treinar o BMD transformamos os dados originais do *stacking* criando novos rótulos e representações com MFs a partir das predições e dos dados das bases. Nesta transformação, definimos como rótulo para um documento o modelo base que acertou a predição da classe do documento com maior confiança.

Analisamos a seguir a nova distribuição de rótulos comparada com a distribuições de predição do BMD. Inicialmente, destacamos o desequilíbrio existente nos dados utilizados para treinar o BMD. Como podemos ver na Figura 6.1, para todas as bases de dados, a transformação nos dados criou meta-dados desbalanceados, principalmente para

Tabela 6.4: Efetividade do BMD base do *stacking*.

Tipo Prob.		ACM	20NG	WebKB	AGNews	IMDb	Sogou	Média
Sem Calibrar	Precisão	26.74	38.27	30.83	47.23	28.05	49.28	36.73
	Recall	23.39	24.42	26.59	33.38	21.67	34.82	27.38
	MacroF1	20.81	25.60	24.04	33.65	20.07	36.86	26.84
Calibrado	Precisão	32.79	36.23	38.43	36.43	25.87	34.01	33.96
	Recall	27.23	28.78	30.01	29.75	19.44	25.54	26.79
	MacroF1	25.95	26.81	28.55	26.22	16.64	24.66	24.81

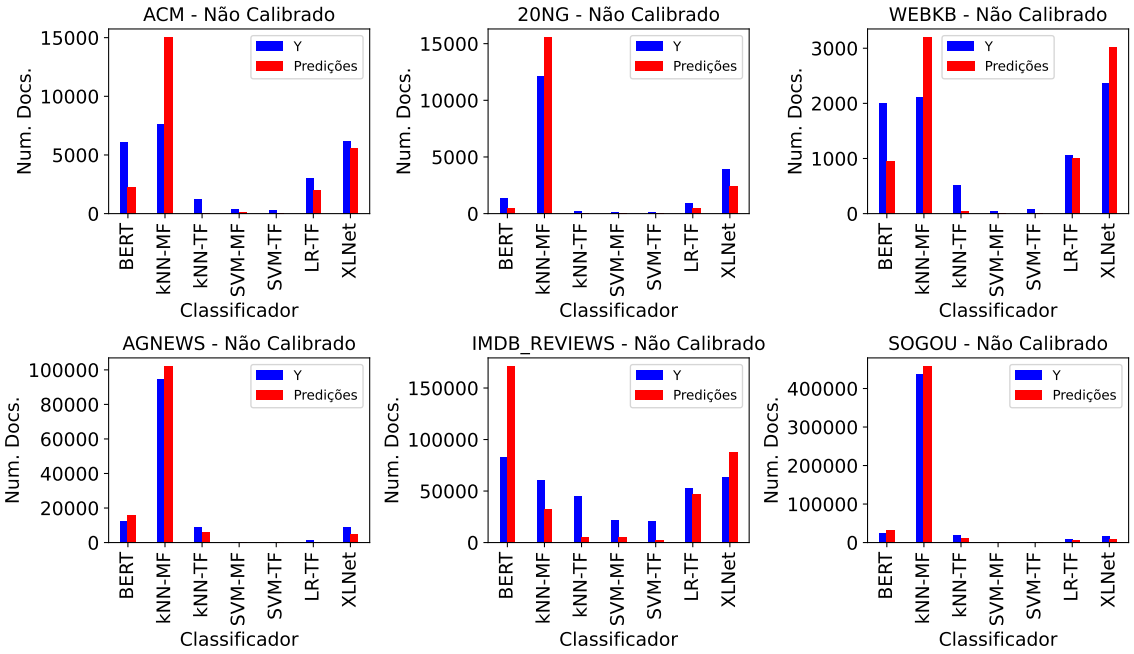

Figura 6.1: Distribuição de classes e predições na seleção do melhor modelo com probabilidades sem calibrar. Em azul temos a quantidade de documentos em que um classificador foi indicado como o ideal e em vermelho tracejado a quantidade de vezes que o BMD indicou o classificador como o ideal.

bases originalmente balanceadas em CAD, como a 20NG, AGNews e Sogou. Podemos ver que no modelo kNN-MF concentra-se grande volume de rótulos para estas bases (64,51% 20NG, 74,22% AGNews e 85,87% Sogou). Podemos ver também que este comportamento enviesa o seletor de modelo base de tal modo que o modelo base kNN-MF concentra a maioria das predições do BMD (82,35% 20NG, 79,81% AGNews e 89,54% Sogou).

Em contraste, para as bases originalmente mais desbalanceadas a transformação dos dados produziu meta-dados com distribuições de rótulos mais balanceadas. Como podemos ver, para as bases ACM WebKB e IMDb os rótulos estão menos concentrados em um único modelo base, e os modelos BERT, LR-TF e XLNet começam a ficar mais proeminentes, junto com o kNN-MF. Além disso, os modelos SVM-MF e SMV-TF começam a concentrar volumes mais expressivos de rótulos (o que não ocorreu para as bases balanceadas). Esta diferença no equilíbrio da base, porém, não impediu que o BMD ficasse enviesado em direção a um modelo base.

Podemos ver que para as bases ACM e WebKB o kNN-MF foi identificado como o melhor modelo base em 60,21% e 38,91% dos documentos contra 30,57% e 25,70% em que este era de fato o modelos mais adequado, respectivamente. Comportamento similar é observado sobre a base IMDb com o modelo base BERT que é identificado como o melhor modelo em 48,99% dos documentos quando na verdade é o mais adequado em apenas 23,87% dos documentos. Esta última observação faz sentido, pois dado que o BERT é um modelo contextual e a base IMDb é uma base de dados de avaliação de filmes por

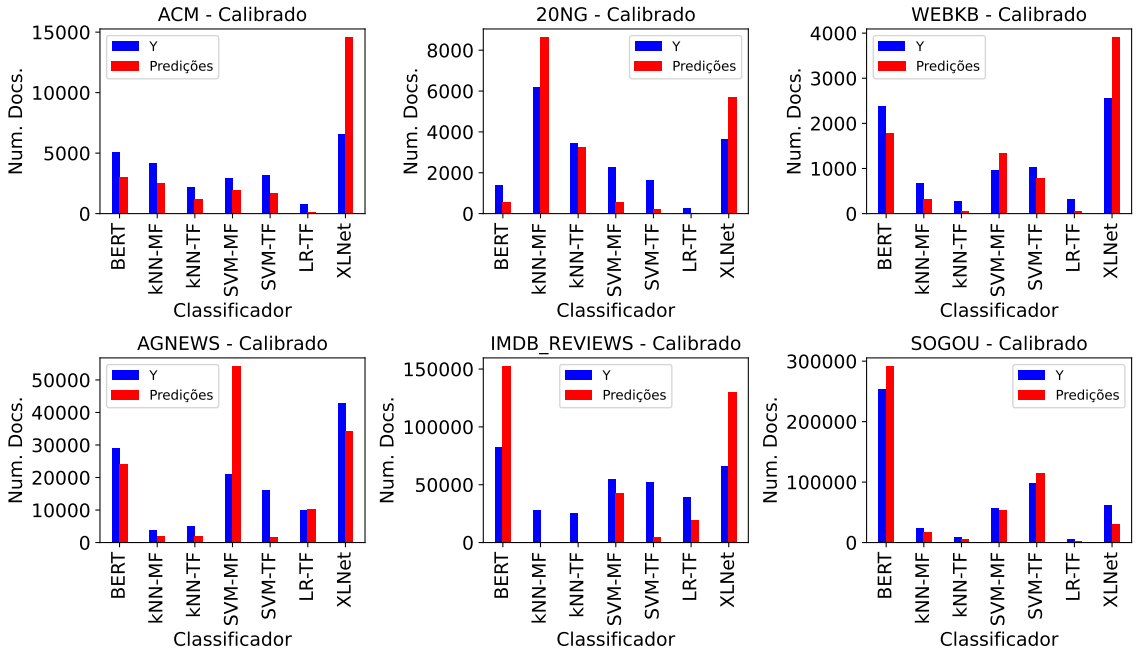

Figura 6.2: Distribuição de classes e predições na seleção do melhor modelo com probabilidades calibradas. Em azul temos a quantidade de documentos em que um classificador foi indicado como o ideal e em vermelho tracejado a quantidade de vezes que o BMD indicou o classificador como o ideal.

usuários, o contexto é de suma importância, logo permitindo que o BERT seja um modelo mais assertivo e confiante consistindo no modelo base mais real adequado e sendo também identificado mais vezes como o mais adequado para predição nesta base.

Uma justificativa para o desbalanceamento nos meta-dados utilizados pelo BMD base é que bases originalmente mais desbalanceadas em CAD são mais desafiadoras para os modelos base, uma vez que as bordas de separação entre as classes não são tão bem definidas. Este comportamento consequentemente reduz a confiança nas predições dos modelos levando que suas distribuições sejam menos enviesadas em direção a um único modelo específico "super confiante".

Considerando a efetividade do BMD medida via MacroF1, a aplicação de calibragem foi mais prejudicial em média. Porém, a calibragem ajudou a melhorar o desbalanceamento dos meta-dados em relação ao seu par não calibrado. Probabilidades calibradas ajudaram a produzir meta-dados mais calibrados para o BMD.

Na Figura 6.2 temos a distribuição de rótulos e predições gerados sobre probabilidades calibradas. Primeiro detalhe a se notar é a redução da concentração de documentos no modelo kNN-MF sobre as diferentes bases de dados. Apesar de ser o modelo definido como mais adequado para a base 20NG (32,89%) e mais indicado pelo BMD (45,68%), este modelo não exerce mais dominância sobre as bases AGNews e Sogou (que agora são dominadas pelo modelo BERT com 22,82% e 49,65% dos documentos, respectivamente). Além disso, os modelos BERT, XLNet e SVM-MF começam a concentrar maior parte das

predições do BMD.

O modelo BERT domina as bases IMDb e Sogou com 43.63% e 57,03% das predições. Além de manter volume considerável nas bases WebKB (21,72%) e AGNews (18,81%) dos documentos. Já o modelo SVM-MF aumenta sua participação em todas as bases e domina a base AGNews, recebendo 42,38% das predições quando na verdade é o modelo ideal em apenas 16,42% dos documentos.

Este comportamento é interessante dado que modelos com base em SVM são geralmente pouco confiantes. Logo a calibragem permite ajustar as probabilidades dos modelos baseados em SVM aumentando sua participação na estratégia, consequentemente contribuindo para a diversidade do *ensemble*. O modelo XLNet torna-se o mais dominante junto a BERT. Para as bases ACM, WebKB e IMDb o modelo base XLNet recebe 58,38%, 47,53% e 37,27% das predições do BMD, respectivamente. A proeminência dos modelos BERT e XLNet é fortemente causada pela combinação da calibragem de modelos profundos com *Temperature Scaling* e calibragem de modelos tradicionais com *Isotonic*. *Temperature Scaling* provoca transformações mais suaves (e nunca mudam as predições originais) afetando pouco a super confiança destes modelos. Já a estratégia *Isotonic* provoca alterações mais profundas (podendo alterar as predições originais dos modelos base) reduzindo mais bruscamente a confiança dos modelos base.

Por fim, comparando os resultados da técnica com probabilidades calibradas e não calibradas, podemos ver que a diferença na efetividade entre ambos consiste na diferença de desbalanceamento. Dado que a produção de meta-dados para o BMD produz grandes volumes de rótulos de modelo mais adequado concentrados em um único classificador, o BMD tem seu trabalho facilitado ficando enviesado em escolher sempre um modelo específico (neste caso o kNN-Mf) aumentando a probabilidade de sucesso. Porém, a aplicação de técnicas de calibragem de probabilidade podem ser benéficas para a estratégia Best Model na perspectiva que esta ameniza o desbalanceamento dos dados e reduz o viés do BMD em direção a um modelo base. Isso contribui para a diversidade na escolhas exercidas pelo BMD (contribuindo para melhor evolução da técnica), o que pode ser benéfico para o ensemble como apresentados em outros trabalhos da literatura na seção 3.2.

6.3 Detecção e Tratamento de Documentos Difíceis

Nesta seção apresentamos os resultados obtidos com a estratégia Hard Docs. Na seção 6.3.1 apresentamos o desempenho da estratégia na classificação de documentos comparando com os resultados obtidos com o *stacking* tradicional. Na seção 6.3.2 apresenta-

Tabela 6.5: Desempenho geral (MacroF1) da estratégia Hard Docs. Avaliamos o desempenho da estratégia utilizando probabilidades calibradas (Calib.) e sem calibrar (Normal). Para facilitar a comparação, destacamos em cinza os resultados obtidos com o *stacking* tradicional. Referenciamos o limite superior da estratégia adicionando o prefixo "Upp." ao seu nome. Destacamos com triângulos verdes e vermelhos invertidos os ganhos e perdas estatísticos comparando cada estratégia com o *stacking* tradicional, respectivamente. Empates estatísticos destacamos com um círculo amarelo. Utilizamos *teste t student* com 95% de confiança.

	20NG		ACM		WEBKB	
	Normal	Calib.	Normal	Calib.	Normal	Calib.
Stacking	92.24	92.24 ●	73.92	73.76 ●	82.92	83.1 ●
Upp. Hard Docs	92.04 ●	91.86 ●	74.13 ●	73.84 ●	84.17 ▲	84.35 ●
Hard Docs	90.38 ▼	90.49 ▼	72.88 ▼	72.5 ▼	82.26 ●	80.58 ▼
	AGNews		IMDb		Sogou	
	Normal	Calib.	Normal	Calib.	Normal	Calib.
Stacking	94.9	94.92 ●	35.27	35.22 ●	96.87	96.88 ●
Upp. Hard Docs	95.82 ▲	95.82 ▲	34.23 ▼	34.42 ▼	97.46 ▲	97.27 ▲
Hard Docs	93.2 ▼	93.13 ▼	35.1 ▼	35.15 ●	94.81 ▼	95.81 ▼

mos um estudo sobre a importância do tratamento dos documentos difíceis e a efetividade na estatégia Hard Docs na detecção destes.

6.3.1 Comparação Hard Docs e Stacking

A estratégia **Hard Docs**, diferente das demais estratégias desse *framework* apresenta menor potencial em seu *Upperbound*. Como mostrado na Tabela 6.5 o *Upperbound* da estratégia supera o *stacking* tradicional nas bases WebKB (1,5% sem calibragem) AGNews (0,96% sem e com calibragem) e Sogou (0,6% sem calibragem e 0,41% com calibragem). Para as demais bases de dados o *Upperbound* da estratégia empata estatisticamente exceto para a base de dados IMDb, onde o *Upperbound* obtém resultado inferior ao *stacking* (ocasionado devido a falhas do classificador especialista em documentos fáceis). **Mesmo para as bases onde o *Upperbound* supera o *stacking* ainda não conseguimos evidência que de a estratégia Hard Docs é capaz de superar o *stacking*. Desta forma, os resultados obtidos pela estratégia respondem negativamente a terceira Questão de Pesquisa QP3 deste trabalho**.

O baixo potencial do *Upperbound* e da estratégia Hard Docs evidencia a necessidade de melhorar a efetividade do meta-classificador de documentos difíceis, tanto para documentos difíceis quanto para fáceis. Apesar do baixo potencial apresentado pelo *Upperbound* da estratégia Hard Docs, está estratégia ainda obteve valores próximos ao do

stacking em todas as bases de dados (2,17% abaixo no pior cenário para base Sogou sem calibragem). Importante destacar que diferente das demais estratégias para base IMDb os resultados ficaram próximos ao *stacking*, incluindo um empate estatístico na versão calibrada da estratégia. Esta é uma característica positiva da estratégia, pois apesar de não obter resultados superiores, apresenta valores próximos ao do *stacking* nos diferentes cenários.

Na próxima seção analisamos a efetividade dos modelos de detecção de documentos difíceis.

6.3.2 Efetividade na Detecção de Documentos Difíceis

Efetividade na detecção de documentos difíceis é muito importante para garantia da eficácia da estratégia **Hard Docs**. Como previamente apresentado na tabela 6.5 (seção 6.3.1) a simulação do detector de documentos difíceis ideal (*Upperbound*) supera os resultados do *stacking* para algumas bases de dados. Apesar da construção de um detector de documentos difíceis ideal ser uma tarefa praticamente intangível, investigamos o quão distante os detectores desenvolvidos neste trabalho estão de suas versões ideais.

Documentos difíceis são naturalmente desafiadores para o meta-classificador. Como mostrado na Tabela 6.6 exceto para a base de dados IMDb, estes documentos representam cerca de 20% ou menos da base de dados. Além disso, estes documentos são responsáveis pela maioria das falhas do meta-classificador (concentrando sempre mais de 82% das falhas em todas as bases de dados - permanecendo alto mesmo para base IMDb). Este comportamento nos mostra a importância de tratar tais documentos com precisão e o desafio inerente ao desbalanceamento entre documentos fáceis e difíceis.

Tabela 6.6: Estatísticas sobre documentos difíceis. Nesta tabela apresentamos por base de dados a quantidade de documentos fáceis, difíceis (com valor absoluto e percentual), quantidade de falhas do meta-classificador e quantidade de falhas do meta-classificador sobre documentos difíceis (valor absoluto e o percentual em relação ao total de falhas).

Bases de Dados	#Fáceis	#Difíceis	#Falhas do Meta-Classificador	#Falhas em Docs. Difíceis
ACM	19805	5092 (20,45%)	4276	3869 (90,48%)
20NG	17008	1838 (9,75%)	1419	1223 (86,18%)
WebKB	6804	1395 (17,01%)	974	803 (82,44%)
AGNews	118875	8725 (6,83%)	6506	5556 (85,39%)
IMDb	81009	267406 (76,74%)	216087	199989 (92,55%)
Sogou	488026	21974 (4,30%)	15962	14018 (87,82%)

Tabela 6.7: Desempenho dos modelos de detecção de documentos difíceis (DD). Reportamos a precisão, *recall* (cobertura) e MacroF1 obtidos na identificação de documentos difíceis.

	ACM		20NG		WebKB	
	Normal	Calibrado	Normal	Calibrado	Normal	Calibrado
Precisão	54.5	53.05	55.31	59.22	65.4	60.47
Recall	76.18	76.24	78.59	80.79	68.23	70.09
MacroF1	63.52	62.55	64.81	68.2	66.63	64.76
	AGNews		IMDb		Sogou	
	Normal	Calibrado	Normal	Calibrado	Normal	Calibrado
Precisão	32.69	31.66	99.5	98.27	28.02	28.7
Recall	80.85	84.31	67.44	62.66	86.67	87.35
MacroF1	46.51	46	80.39	76.53	42.33	43.18

Na tabela 6.7 reportamos a precisão, *recall* (cobertura) e a MacroF1 de cada modelo de detecção de documentos difíceis para cada base de dados. De modo geral o primeiro ponto a se notar é que a calibragem dos modelos impactou ligeiramente no desempenho dos detectores, provocando alterações na precisão das bases 20NG (55.31 para 59.22) e WebKB (65.4 para 60.47) e no *recall* (67.44 para 62.99) da base de dados IMDb. No entanto, exceto para as bases 20NG e IMDb não houveram significativas diferenças na MacroF1. Este comportamento é esperado uma vez que as MFs propostas neste trabalho são pouco sensíveis a distorções nas distribuições de probabilidade.

De forma geral, o *recall* dos detectores é superior à precisão para todas as bases de dados, exceto para base IMDb. Um dos desafios na detecção de documentos difíceis como descrito previamente é a diferença entre o volume de documentos difíceis e fáceis, sendo de forma geral, o número de documentos difíceis representando menos que 20% da base de dados. No entanto, para base IMDb, a base mais desafiadora deste trabalho, o número de documentos fáceis corresponde a 23,26% da base. Diferente das demais bases, onde aparentemente mesmo com a aplicação de *oversampling* a precisão foi prejudicada, na base IMDb obtivemos elevada precisão (99,5 normal e 98,27 calibrado). Desta forma, pode-se perceber que a precisão dos DDs é o ponto que mais precisa de esforços para o aprimoramento dessa estratégia. Além disso, para maioria das bases a baixa precisão parece estar correlacionada com o baixo volume de documentos difíceis em relação aos fáceis, sendo que para IMDb, única base que esta relação é inversa, a precisão obtida foi superior ao *recall*.

Capítulo 7

Conclusão e Trabalhos Futuros

Neste trabalho apresentamos um estudo sobre previsão e tratamento de falhas de modelos base de *stacking*. Mais especificamente introduzimos um novo *framework* PTFS com três estratégias (Error Detection, Best Model e Hard Docs) com foco em aumentar a efetividade de modelos de *stacking* em CAD através da previsão e tratamento de Falhas de Predição de modelos base do *stacking* de forma dinâmica.

Na estratégia Error Detection, treinamos para cada modelo base um estimador de falhas. O estimador de falhas, ao prever uma falha do modelo base anula as predições do modelo base impedindo que estas exerçam influência na decisão do meta-classificador. Em Best Model nós focamos em identificar o modelo base mais adequado em meio aos modelos do *stacking* para fazer a predição da classe do documento. Nossa terceira estratégia, Hard Docs focamos em documentos onde poucos modelos modelo base acertam (documentos difíceis). Nesta estratégia criamos um *pipeline* em dois passos onde no primeiro identificamos documentos difíceis e no segundo treinamos um meta-classificador especialista para classificar os documentos difíceis.

Nosso arcabouço experimental é composto por sete modelos (dois modelos profundos e cinco modelos de aprendizado de máquina tradicionais). Para todos os sete modelos base do *stacking* realizamos a avaliação das três estratégias contidas no *framework* PTFS. Além disso, realizamos os experimentos previamente citados novamente porém, adicionando um processo de calibragem de probabilidades para aproximar a distribuição de probabilidades dos modelos base da real distribuição de probabilidades das classes. Os experimentos aqui realizados nos permite investigar o quanto as Falhas de Predição de modelos base impactam no desempenho de meta-classificadores e propor novas alternativas para evolução de modelos de *stacking* através da detecção e tratamento automático de Falhas de Predição.

Por meio das três estratégias de nosso *framework*, simulamos, para cada base de dados diferentes, *Upperbounds* (cenários simulando o tratamento de falhas ideal ou seja limite superior da estratégia das estratégias do *framework*). Com a identificação do *Upperbound* mostramos aumento considerável na efetividade do *stacking* com destaque para as bases de dados ACM, WebKB e IMDb. Sobre estas bases a estratégia Error Detection revelou grande potencial de melhoria de efetividade em relação ao *stacking* tradicional

com margens de MacroF1 superiores à 17% ACM, 13% WebKB e 67% IMDb tanto para suas versões não calibrada e calibradas. Comportamento similar foi apresentado com a estratégia Best Model com margens de potencial superiores à 16% ACM, 13% WebKB e 73% IMDb. A estratégia Hard Docs diferentes das demais apresentou menos potencial de evolução.

Apesar do elevado potencial apresentado nos diferentes cenários as estratégias do nosso *framework*, as implementações destas estratégias ainda não foram capazes de superar o *stacking* usual. Ainda assim, as estratégias se mostraram competitivas obtendo níveis de efetividade muito próximos ao *stacking* sem diferença significativa (empate estatístico) em grande parte das bases de dados.

Além da análise do potencial das estratégias (cenário ideal e implementação) realizamos análises do desempenho das nossas estratégias na detecção e tratamento de falhas. Nossas estratégias mostram resultados promissores na previsão de falhas. Com os modelos de previsão de falhas na estratégia Error Detection conseguimos antecipar falhas de modelos com precisão média de 75,98%, mínima de 60,15% e máxima de 90,17% e *recall* médio de 67,81% com mínimo de 31,58% e máximo de 97,5%. Nossas estratégias mostraram mais dificuldade em antecipar falhas de modelos profundos do que de modelos tradicionais o que acreditamos ser devido a complexidade dos modelos.

Na escolha do melhor modelo com a estratégia Best Model mostramos o quão desafiador é identificar o melhor modelo em um *ensemble*, onde obtivemos média de 31,61 e 28,94 pontos de MacroF1 na detecção do melhor modelo com probabilidades não calibradas e calibradas, respectivamente. Apesar do baixo valor médio de MacroF1 na detecção do melhor modelo a efetividade da estratégia ficou próxima ao *stacking* empatando em grande parte das bases de dados. Verificamos também que a distribuição de rótulos ligeiramente influencia no resultado da seleção do melhor modelo, sendo que probabilidades calibradas proporcionam distribuições de rótulos mais distribuídas em relação entre os modelos base, porém, não necessariamente contribui para a efetividade na seleção do melhor modelo.

Por fim, analisamos o desempenho da estratégia de identificação de documentos difíceis utilizada na estratégia Hard Docs. Documentos difíceis correspondem a um pequeno volume das bases de dados (menos que 20%, exceto IMDb), porém concentram grande volume das falhas dos meta-classificadores mostrando a importância de tratar estes documentos a parte (mais que 80%). Nossos experimentos mostram resultados promissores na detecção de documentos difíceis obtendo média de 60,69 e 60,20 pontos de MacroF1 com probabilidades não calibradas e calibradas, respectivamente.

Com base no potencial mostrado pelos Upperbounds, principalmente das estratégias Error Detection e Best Model, este trabalho apresentou formas alternativas para melhoria da efetividade dos modelos de classificação baseados em Stacking. De modo geral, todas as estratégias deste trabalho dependem das MFs propostas para treinar os modelos de previsão de falha (Error Detection), seleção de melhor modelo (Best Model) e identificação

de documentos difíceis (Hard Docs). Portanto, de forma geral, como trabalho futuro vislumbramos o desenvolvimento de novas MFs como um dos caminhos para evolução da previsão de falhas e consequentemente das estratégias do *framework*.

Além da introdução de novas MFs, as estratégias aqui propostas podem se beneficiar da utilização de zonas de confiança (zonas de competência)[21, 35, 49]. Zonas de confiança consistem em conjuntos de dados de validação utilizados para seleção de modelos base em *ensembles*. Através destes conjuntos de dados temos indícios de como um modelo se comportará em relação a um novo documento testando-o em documentos similares. Por exemplo, para aprimorar a estratégia Best Model podemos utilizar zonas de competência para auxiliar a identificar o modelo mais adequado. Basicamente, podemos testar cada modelo base em uma zona de competência (conjunto de validação com documentos próximos ao documento a ser rotulado) e utilizar a informação do desempenho do modelo na zona de confiança para ponderar a decisão do seletor de melhor modelo. A aplicação de zonas de confiança é também facilmente extensível para previsão de falhas e identificação de documentos difíceis, necessitando apenas criar as zonas de confiança no formato adequado para avaliação do previsor de falhas e do detector de documentos difíceis.

Outro foco de melhoria principalmente para estratégia Error Detection consiste na utilização de modelos de *adaptive learning* [37, 42] para seleção de modelos base. Estas estratégias consistem em modelos equipados com dispositivos de atenção especializados em seleção dinâmica de modelos. Os pesos da camada de atenção refletem a importância de cada modelo de acordo com o documento. A utilização de abordagens de *adaptive learning* engloba o pipeline completo da estratégia Error Detection em uma única arquitetura neural. Sugerimos que a alimentação destas arquiteturas com as MFs propostas neste trabalho como representação ou parte da representação dos documentos pode auxiliar o modelo no estabelecimento de pesos mais coerentes para cada modelo base do *ensemble*, uma vez que as MFs propostas possuem alto poder preditivo focado em previsão de falha. Além disso, estas arquiteturas facilitam o desenvolvimento de novas funções de perda (*loss functions*) mais adequadas para o problema.

Referências

[1] S Abarna, JI Sheeba, and S Pradeep Devaneyan. An ensemble model for idioms and literal text classification using knowledge-enabled bert in deep learning. *Measurement: Sensors*, 24:100434, 2022.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[3] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409, 09 2014.

[5] Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E. Schapire. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651 – 1686, 1998.

[6] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, jul 2015.

[7] Nitin Bhatia and Vandana. Survey of nearest neighbor techniques. *International Journal of Computer Science and Information Security*, 8, 07 2010.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2017.

[10] Bernhard Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifier. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 5, 08 1996.

[11] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 1 2001.

[12] Raphael Campos, Sérgio Canuto, Thiago Salles, Clebson C.A. de Sá, and Marcos André Gonçalves. 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*, SIGIR '17, page 105–114, New York, NY, USA, 2017. Association for Computing Machinery.

[13] Sérgio Canuto, Marcos Gonçalves, Wisllay Santos, Thierson Rosa, and Wellington Martins. An efficient and scalable metafeature-based document classification approach based on massively parallel computing. SIGIR '15, page 333–342, New York, NY, USA, 2015. Association for Computing Machinery.

[14] Sergio Canuto, Thiago Salles, Marcos André Gonçalves, Leonardo Rocha, Gabriel Ramos, Luiz Gonçalves, Thierson Rosa, and Wellington Martins. On efficient meta-level features for effective text classification. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, page 1709–1718, New York, NY, USA, 2014. Association for Computing Machinery.

[15] Sergio Canuto, Thiago Salles, Thierson C. Rosa, and Marcos A. Gonçalves. Similarity-based synthetic document representations for meta-feature generation in text classification. SIGIR'19, page 355–364, New York, NY, USA, 2019. Association for Computing Machinery.

[16] Sérgio Canuto, Daniel Xavier Sousa, Marcos André Gonçalves, and Thierson Couto Rosa. A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2242–2256, 2018.

[17] Jonnathan Carvalho and Alexandre Plastino. On the evaluation and combination of state-of-the-art features in twitter sentiment analysis. *Artificial Intelligence Review*, 54(3):1887–1936, March 1 2021.

[18] William Cavnar and John Trenkle. N-gram-based text categorization. *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, 05 2001.

[19] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[21] Rafael M.O. Cruz, Dayvid V.R. Oliveira, George D.C. Cavalcanti, and Robert Sabourin. Fire-des++: Enhanced online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, 85:149–160, 2019.

[22] Washington Cunha, Vítor Mangaravite, Christian Gomes, Sérgio Canuto, Elaine Resende, Cecilia Nascimento, Felipe Viegas, Celso França, Wellington Santos Martins, Jussara M. Almeida, Thierson Rosa, Leonardo Rocha, and Marcos André Gonçalves. On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *Information Processing & Management*, 58(3):102481, 2021.

[23] Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. *ArXiv*, abs/2003.07892, 2020.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[25] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 193–202, New York, NY, USA, 2014. Association for Computing Machinery.

[26] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[27] Weimin Ding and Shengli Wu. A cross-entropy based stacking method in ensemble learning. *Journal of Intelligent & Fuzzy Systems*, 39:1–12, 07 2020.

[28] Sašo Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54:255–273, 03 2004.

[29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, jun 2008.

[30] Christian Gomes, Marcos Goncalves, Leonardo Rocha, and Sergio Canuto. On the cost-effectiveness of stacking of neural and non-neural methods for text classification: Scenarios and performance prediction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4003–4014, Online, August 2021. Association for Computational Linguistics.

[31] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org, 2017.

[32] Huaping Guo, Hongbing Liu, Ran Li, Changan Wu, Yibo Guo, and Mingliang Xu. Margin & diversity based ordering ensemble pruning. *Neurocomputing*, 275:237–246, 2018.

[33] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[34] Hugging Face. Hugging Face. https://huggingface.co/. Accessed: March 27th.

[35] Wen hui Hou, Xiao kang Wang, Hong yu Zhang, Jian qiang Wang, and Lin Li. A novel dynamic ensemble selection classifier for an imbalanced data set: An application for credit risk assessment. *Knowledge-Based Systems*, 208:106462, 2020.

[36] Ammar Ismael Kadhim. Survey on supervised machine learning techniques for automatic text classification. *Artif. Intell. Rev.*, 52(1):273–292, jun 2019.

[37] Atharva Kulkarni, Amey Hengle, and Rutuja Udyawar. An attention ensemble approach for efficient text classification of Indian languages. In Dipti Misra Sharma, Asif Ekbal, Karunesh Arora, Sudip Kumar Naskar, Dipankar Ganguly, Sobha L, Radhika Mamidi, Sunita Arora, Pruthwik Mishra, and Vandan Mujadia, editors, *Proceedings of the 17th International Conference on Natural Language Processing (ICON): TechDOfication 2020 Shared Task*, pages 40–46, Patna, India, December 2020. NLP Association of India (NLPAI).

[38] Alexandre Lacoste, H. Larochelle, Mario Marchand, and Francois Laviolette. Agnostic bayesian learning of ensembles. *31st International Conference on Machine Learning, ICML 2014*, 2:934–943, 01 2014.

[39] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Trans. Intell. Syst. Technol.*, 13(2), apr 2022.

[40] T S Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228, 09 2000.

[41] Hongzhi Liu, Yingpeng Du, and Zhonghai Wu. Aem: Attentional ensemble model for personalized classifier weight learning. *Pattern Recognition*, 96:106976, 2019.

[42] Hongzhi Liu, Yingpeng Du, and Zhonghai Wu. Generalized ambiguity decomposition for ranking ensemble learning. *Journal of Machine Learning Research*, 23(88):1–36, 2022.

[43] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[44] Ibomoiye Domor Mienye and Yanxia Sun. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, 10:99129–99149, 2022.

[45] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[46] Ammar Mohammed and Rania Kora. An effective ensemble deep learning framework for text classification. *Journal of King Saud University - Computer and Information Sciences*, 34(10, Part A):8825–8837, 2022.

[47] Tien Thanh Nguyen, Anh Vu Luong, Manh Truong Dang, Alan Wee-Chung Liew, and John McCall. Ensemble selection based on classifier prediction confidence. *Pattern Recognition*, 100:107104, 2020.

[48] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 625–632, New York, NY, USA, 2005. Association for Computing Machinery.

[49] Dayvid V.R. Oliveira, George D.C. Cavalcanti, and Robert Sabourin. Online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, 72:44–58, 2017.

[50] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[51] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics.

[52] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10, 06 2000.

[53] Alice Richardson. Logistic regression: A self-learning text, third edition by david g. kleinbaum, mitchel klein. *International Statistical Review*, 79:296–296, 08 2011.

[54] Leonardo Rocha, Gabriel Ramos, Rodrigo Chaves, Rafael Sachetto, Daniel Madeira, Felipe Viegas, Guilherme Andrade, Sérgio Daniel, Marcos Gonçalves, and Renato Ferreira. G-knn: An efficient document classification algorithm for sparse datasets on gpus using knn. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, page 1335–1338, New York, NY, USA, 2015. Association for Computing Machinery.

[55] Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Vote-boosting ensembles. *Pattern Recognition*, 83:119–133, 2018.

[56] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[57] Welton Santos, Washington Cunha, Celso França, Guilherme Fonseca, Sergio Canuto, Leonardo Rocha, and Marcos Gonçalves. Uma metodologia para tratamento do viés da maioria em modelos de stacking via identificação de documentos difíceis. In *Anais do XXXVIII Simpósio Brasileiro de Bancos de Dados*, pages 408–413, Porto Alegre, RS, Brasil, 2023. SBC.

[58] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[59] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? *ArXiv*, abs/1905.05583, 2019.

[60] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2015.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[62] Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. Extending multilingual BERT to low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online, November 2020. Association for Computational Linguistics.

[63] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[64] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[65] Xu-Cheng Yin, Kaizhu Huang, Chun Yang, and Hong-Wei Hao. Convex ensemble learning with sparsity and diversity. *Information Fusion*, 20:49–59, 2014.

[66] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[67] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1):239–263, 2002.