

Projeto Integrado em Mestrado de Matemática e Computação

Sentiment Analysis of Text

Relatório Prático

Ana Cláudia Andrade de Oliveira (PG55613)
Beatriz Fernandes Marques (PG57743)
Helder Amorim Fiúza Fernandes (PG58297)
Matilde Galhetas Domingues (A98982)

Orientador(es):

Prof. Dr. Gaspar José Brandão Queirós Azevedo Machado
Profa. Dra. Sofia Oliveira Lopes

Ano Letivo 2024/2025

Conteúdo

Lista de Figuras	v
Lista de Tabelas	vi
Introdução	1
1 Fundamentos & Datasets Utilizados	2
1.1 Objetivos	2
1.2 Datasets Utilizados	2
1.2.1 Sentiment	2
1.2.2 Emotion	3
1.3 Classificação Binária vs. Classificação Multiclasse	3
1.3.1 Classificação Binária	4
1.3.2 Classificação Multiclasse	4
1.3.3 Principais Diferenças	5
2 Pré-Processamento & Feature Engineering	6
2.1 Objetivos	6
2.2 Configuração do Ambiente de Desenvolvimento	6
2.3 Pré-Processamento dos Dados	7
2.3.1 Técnicas Básicas de Pré-Processamento	7
2.3.2 Técnicas Avançadas de Pré-Processamento	9
2.3.2.1 Stemming vs. Lemmatization	9
2.3.2.2 Word Tokenization vs. Sentence Tokenization	9
2.3.2.3 Tratamento de Negações	10
2.3.2.4 Normalização de Sinônimos	11
2.3.3 Pipeline Final de Pré-Processamento para ambos os Datasets	11
2.3.3.1 Ordem da Pipeline Final	11
2.3.3.2 Resultados Finais do Pré-Processamento	12
2.4 Engenharia de Atributos	13
2.4.1 Técnicas de Representação de Texto	13
2.4.1.1 Bag of Words (BoW)	13
2.4.1.2 N-grams	14
2.4.1.3 TF-IDF (<i>Term Frequency-Inverse Document Frequency</i>)	15
2.4.2 Word Embeddings	16
2.4.2.1 Word2Vec	16
2.4.2.2 GloVe	17
2.4.2.3 FastText	18

3 Seleção de Modelos & Combinações de Parâmetros	19
3.1 Objetivos	19
3.2 Seleção dos Modelos a ser Avaliados	19
3.2.1 Random Forest	19
3.2.2 Regressão Logística	20
3.2.3 Support Vector Machine (SVM)	20
3.3 Escolha das Combinações de Atributos	20
3.3.1 Combinações Testadas para o Dataset de Sentimentos	21
3.3.2 Particularidades para o Dataset de Emoções	22
3.4 Avaliação e Combinações Finais	23
3.4.1 Dataset de Sentimentos	23
3.4.1.1 Ranking de Modelos	23
3.4.1.2 Ranking de Processamento	23
3.4.1.3 Top 3 de Combinações	24
3.4.2 Dataset de Emoções	25
3.4.2.1 Ranking de Modelos	25
3.4.2.2 Ranking de Processamento	25
3.4.2.3 Top 3 de Combinações	25
4 Treino & Otimização dos Modelos	28
4.1 Objetivos	28
4.2 Divisão dos Dados	28
4.3 Treino	28
4.4 BoW + Chi2 + Logistic Regression	29
4.4.1 Arquitetura e Metodologia	29
4.4.1.1 Dataset de Sentimentos	29
4.4.1.2 Dataset de Emoções	30
4.4.2 Processo de Tuning e Otimização	32
4.4.2.1 Dataset de Sentimentos	32
4.4.2.2 Dataset de Emoções	34
4.4.3 Visualizações e Análises Avançadas	36
4.4.3.1 Dataset de Sentimentos	36
4.4.3.2 Dataset de Emoções	40
4.4.4 Resultados Finais e Métricas de Performance	47
4.4.4.1 Dataset de Sentimentos	47
4.4.4.2 Dataset de Emoções	47
4.4.5 Conclusões	48
4.4.5.1 Dataset de Sentimentos	48
4.4.5.2 Dataset de Emoções	48
4.5 TF-IDF _{uni+bi}	49
4.5.1 Arquitetura e Metodologia	49
4.5.1.1 Dataset de Sentimentos	49
4.5.1.2 Dataset de Emoções	50
4.5.2 Processo de Tuning e Otimização	50
4.5.2.1 Dataset de Sentimentos	50
4.5.2.2 Dataset de Emoções:	51
4.5.3 Visualizações e Análises Avançadas	51
4.5.3.1 Dataset de Sentimentos	51

4.5.3.2	Datset de Emoções	56
4.5.4	Resultados	59
4.5.4.1	Dataset de Sentimentos	60
4.5.5	Dataset de Emoções	61
4.5.6	Conclusões	61
4.6	BoW + TF-IDF + GloVe	62
4.6.1	Arquitetura e Metodologia	62
4.6.1.1	Dataset de Sentimentos	62
4.6.1.2	Dataset de Emoções	62
4.6.2	Processo de Tuning e Otimização	63
4.6.3	Dataset de Sentimentos	63
4.6.4	Visualizações e Análises Avançadas	63
4.6.4.1	Dataset de Sentimentos	63
4.6.4.2	Dataset de Emotions	67
4.6.5	Resultados	69
4.6.5.1	Dataset de Sentimentos	69
4.6.5.2	Dataset de Emoções	69
4.6.6	Conclusões	70
5	Desenvolvimento da Interface de Utilizador (UI)	71
5.1	Tecnologias Utilizadas	71
5.2	Arquitetura da Solução	71
5.3	Desafios e Considerações	73
5.4	Testes	73
6	Resultados e Discussão	74
6.1	Objetivos	74
6.2	Comparação entre Datasets	74
7	Conclusão	77

Listas de Figuras

2.1	Pipeline final de pré-processamento e sua implementação em <i>Python</i>	11
2.2	Exemplo <i>BoW</i>	14
2.3	<i>BoW Model Evaluation Accuracy</i> utilizando <i>Naive Bayes</i>	14
3.1	Combinações testadas para o texto processado e respetivos resultados	24
3.2	Combinações testadas para o texto processado e respetivos resultados	26
4.1	Distribuição das classes - dataset de emoções completo	31
4.2	Matriz de Confusão de Validação - Texto Processado	33
4.3	Matriz de Confusão de Teste - Texto Processado	33
4.4	Matriz de Confusão de Validação - Texto Lemmatizado	33
4.5	Matriz de Confusão de Teste - Texto Lemmatizado	33
4.6	Matriz de Confusão de Validação - Texto Processado	35
4.7	Matriz de Confusão de Teste - Texto Processado	35
4.8	Matriz de Confusão de Validação - Texto Lemmatizado	35
4.9	Matriz de Confusão de Teste - Texto Lemmatizado	35
4.10	Análise das Curvas de Treino - Texto Processado	37
4.11	Análise das Curvas de Treino - Texto Lemmatizado	37
4.12	Curvas de Aprendizagem Tradicionais - Texto Processado	38
4.13	Curvas de Aprendizagem Tradicionais - Texto Lemmatizado	38
4.14	Top 20 <i>features</i> positivas mais importantes - texto processado	39
4.15	Top 20 <i>features</i> positivas mais importantes - texto lemmatizado	39
4.16	Top 20 <i>features</i> negativas mais importantes - texto processado	39
4.17	Top 20 <i>features</i> negativas mais importantes - texto lemmatizado	39
4.18	Curvas de Aprendizagem Tradicionais - Texto Processado (Emoções)	40
4.19	Curvas de Validação para o parâmetro C - Texto Processado (Emoções)	40
4.20	Top 10 <i>features</i> positivas para Anger - texto processado	41
4.21	Top 10 <i>features</i> positivas para Anger - texto lemmatizado	41
4.22	Top 10 <i>features</i> negativas para Anger - texto processado	41
4.23	Top 10 <i>features</i> negativas para Anger - texto lemmatizado	41
4.24	Top 10 <i>features</i> positivas para Fear - texto processado	42
4.25	Top 10 <i>features</i> positivas para Fear - texto lemmatizado	42
4.26	Top 10 <i>features</i> negativas para Fear - texto processado	42
4.27	Top 10 <i>features</i> negativas para Fear - texto lemmatizado	42
4.28	Top 10 <i>features</i> positivas para Joy - texto processado	42
4.29	Top 10 <i>features</i> positivas para Joy - texto lemmatizado	42
4.30	Top 10 <i>features</i> negativas para Joy - texto processado	43
4.31	Top 10 <i>features</i> negativas para Joy - texto lemmatizado	43
4.32	Top 10 <i>features</i> positivas para Love - texto processado	43

4.33	Top 10 <i>features</i> positivas para Love - texto lemmatizado	43
4.34	Top 10 <i>features</i> negativas para Love - texto processado	44
4.35	Top 10 <i>features</i> negativas para Love - texto lemmatizado	44
4.36	Top 10 <i>features</i> positivas para Sad - texto processado	44
4.37	Top 10 <i>features</i> positivas para Sad - texto lemmatizado	44
4.38	Top 10 <i>features</i> negativas para Sad - texto processado	45
4.39	Top 10 <i>features</i> negativas para Sad - texto lemmatizado	45
4.40	Top 10 <i>features</i> positivas para Surprise - texto processado	45
4.41	Top 10 <i>features</i> positivas para Surprise - texto lemmatizado	45
4.42	Top 10 <i>features</i> negativas para Surprise - texto processado	46
4.43	Top 10 <i>features</i> negativas para Surprise - texto lemmatizado	46
4.44	Top 20 <i>features</i> globais - texto processado	46
4.45	Top 20 <i>features</i> globais - texto lemmatizado	46
4.46	Análise das Curvas de Treino - Regressão Logística	52
4.47	Análise das Curvas de Treino - SVM	52
4.48	Curvas ROC e <i>Precision-Recall</i> para o modelo de Regressão Logística	53
4.49	Curvas ROC e <i>Precision-Recall</i> para o modelo de SVM	53
4.50	Top-20 de termos TF-IDF para a Regressão Logística	54
4.51	Matrizes de Confusão de Treino e Teste do modelo de Regressão Logística	55
4.52	Matrizes de Confusão de Treino e Teste do modelo de SVM	55
4.53	Análise das Curvas de Treino - Texto Processado	56
4.54	Análise das Curvas de Treino - Texto Processado e Lematizado	56
4.55	Top-20 de termos TF-IDF para o texto processado	57
4.56	Top-20 de termos TF-IDF para o texto processado e lematizado	58
4.57	Matrizes de Confusão de Treino e de Teste para o texto processado	59
4.58	Matrizes de Confusão de Treino e de Teste para o texto processado e lematizado	59
4.59	Análise das Curvas de Treino - Texto Processado	64
4.60	Análise das Curvas de Treino - Texto Processado e Lematizado	64
4.61	Curvas ROC e <i>Precision-Recall</i> para o texto processado	65
4.62	Curvas ROC e <i>Precision-Recall</i> para o texto processado e lematizado	65
4.63	Top-20 de termos BoW para o texto processado	66
4.64	Top-20 de termos BoW para o texto processado e lematizado	66
4.65	Matrizes de Confusão de Validação e Teste para o texto processado	67
4.66	Matrizes de Confusão de Validação e Teste para o texto processado e lematizado	67
4.67	Curvas ROC por classe - Texto Processado (Validação)	68
4.68	Matriz de Confusão - Texto Processado (Validação)	68
5.1	Visualização da Página	72

Lista de Tabelas

2.1	Métricas de avaliação do impacto do pré-processamento nos <i>datasets</i> de sentimentos e emoções	12
3.1	Top 3 Combinações de Atributos para o Dataset de Sentimentos	25
3.2	Top 3 Combinações de Atributos para o Dataset de Emoções	26
4.1	Comparação entre Texto Processado e Texto Lemmatizado	33
4.2	Comparação entre Texto Processado e Texto Lemmatizado - Dataset de Emoções	35
4.3	Hiperparâmetros utilizados no vetor TF-IDF	50
4.4	Hiperparâmetros para o modelo de Regressão Logística	50
4.5	Hiperparâmetros para o modelo SVM	51
4.6	Hiperparâmetros para o modelo de Regressão Logística	51
4.7	Melhores hiperparâmetros para o <i>dataset</i> de sentimentos	60
4.8	Métricas de desempenho para Regressão Logística no <i>dataset</i> de sentimentos	60
4.9	Métricas de desempenho para SVM no <i>dataset</i> de sentimentos	60
4.10	Métricas de desempenho para texto processado no <i>dataset</i> de emoções	61
4.11	Métricas de desempenho para texto lematizado no <i>dataset</i> de emoções	61
4.12	Descrição dos Parâmetros	63
4.13	Métricas de desempenho para texto processado	69
4.14	Métricas de desempenho para texto processado e lematizado	69
4.15	Métricas de desempenho para texto processado - Dataset de Emoções	69
4.16	Métricas de desempenho para texto processado e lematizado - Dataset de Emoções	69
4.17	Métricas de desempenho por classe emocional - Texto Processado (Validação) . . .	70

Introdução

A análise de sentimentos é uma área do Processamento de Linguagem Natural (NLP) que visa identificar e interpretar emoções em textos, sendo amplamente aplicada em domínios como redes sociais, avaliações de produtos e atendimento ao cliente. O presente relatório descreve o desenvolvimento de um sistema de *Sentiment Analysis of Text*, desde a aquisição e preparação dos dados até a construção, otimização e avaliação de modelos de classificação.

O projeto foi organizado em quatro fases principais, distribuídas ao longo de doze semanas. Na primeira fase, foram estabelecidas as bases do trabalho, incluindo a configuração do ambiente de desenvolvimento, a definição dos objetivos e a recolha e exploração inicial dos dados. Seguiu-se a fase de pré-processamento e engenharia de atributos, onde foram aplicadas técnicas avançadas para transformar os textos em representações adequadas para modelos de machine learning. Na terceira fase, foram treinados e avaliados diferentes modelos de classificação, desde abordagens simples, como *Regressão Logística*, até técnicas mais avançadas, explorando a otimização de hiperparâmetros para melhorar o desempenho. A última fase focou-se no refinamento do modelo, na realização de testes em dados não vistos e na documentação e apresentação dos resultados obtidos.

Este relatório detalha cada uma dessas etapas, descrevendo os métodos utilizados, os desafios encontrados e as soluções implementadas para desenvolver um sistema eficiente de análise de sentimentos.

Capítulo 1

Fundamentos & Datasets Utilizados

1.1 Objetivos

Nas primeiras três semanas do projeto, o foco esteve na construção das bases fundamentais para a análise de sentimentos, começando pela introdução ao ambiente de desenvolvimento e ferramentas essenciais, seguida pela aquisição e exploração de dados e, por fim, pela compreensão e pré-processamento inicial dos mesmos.

Na primeira semana, foram estabelecidos os objetivos do projeto, apresentados os membros da equipa e configurado o ambiente de desenvolvimento, incluindo linguagens, bibliotecas e ferramentas como Python, Git e frameworks para *NLP*. Além disso, foi realizada uma introdução aos conceitos e desafios da análise de sentimentos.

A segunda e terceira semanas centraram-se na recolha de dados adequados para a tarefa, identificando e descarregando *datasets* públicos, como críticas de filmes, avaliações de produtos e publicações em redes sociais. A estrutura e distribuição dos dados foram analisadas e foram geradas visualizações iniciais para melhor compreensão do conjunto de dados.

Este capítulo detalha cada uma destas etapas, descrevendo os métodos utilizados, os desafios encontrados e os resultados obtidos.

1.2 Datasets Utilizados

Para a realização deste projeto prático, foi selecionado o *dataset* “**Sentiment and Emotion Analysis Dataset**”, obtido a partir da plataforma **Kaggle**. Segundo o autor da base de dados, esta foi concebida para apoiar investigadores e praticantes na área do Processamento de Linguagem Natural no desenvolvimento e avaliação de modelos de aprendizagem automática e profunda.

Embora o problema inicialmente proposto para a realização deste projeto se limitasse à classificação binária de sentimentos (positivo vs. negativo), a reduzida dimensão desta amostra — 3.309 frases — levou-nos a considerar também a vertente de classificação multiclasse de emoções. Esta abordagem mais abrangente permite explorar com maior profundidade as capacidades dos modelos e a riqueza emocional presente nos dados.

1.2.1 Sentiment

O subconjunto destinado à análise de sentimentos é composto por 3.309 frases, cada uma rotulada com uma das duas classes:

- **Sentimento Positivo:** 1.679 amostras (50,7%)
- **Sentimento Negativo:** 1.630 amostras (49,3%)

Este subconjunto é adequado para tarefas de classificação de polaridade, como a análise de opiniões, *feedback* de utilizadores ou outras aplicações onde a distinção entre sentimentos positivos e negativos é suficiente.

No entanto, a quantidade de frases deste *dataset* revelou-se demasiado reduzida para permitir o desenvolvimento de modelos robustos e fiáveis. Por exemplo, ao aplicar uma divisão clássica de 70% para treino e 30% para validação e teste, ficaríamos apenas com aproximadamente 2.316 frases para treino e 993 para validação e teste combinados - um volume limitado para captar variações subtis de linguagem.

Para reduzir esta limitação, recorremos ao auxílio do *ChatGPT*, solicitando a geração de novas frases com diferentes níveis de complexidade na análise de sentimento (mantendo sempre a razão de proporção entre as classes), incluindo exemplos ambíguos ou mais desafiantes, como frases com negações (“*not bad*”, “*could be worse*”, entre outras), que ajudam a aumentar a robustez do modelo.

Com esta expansão artificial, o número de amostras foi duplicado, alcançando um total de 6.618 frases. Esta abordagem permitiu não só aumentar a quantidade de dados, mas também melhorar a diversidade linguística e semântica do conjunto, contribuindo para um treino mais eficaz dos modelos de classificação.

- **Sentimento Positivo:** 3.463 amostras (52.3%)
- **Sentimento Negativo:** 3.155 amostras (47.7%)

1.2.2 Emotion

O subconjunto dedicado à análise de emoções é significativamente mais extenso, contendo 422.746 frases, classificadas em seis categorias emocionais distintas:

- **Alegria (Joy):** 143.067 amostras (33.8%)
- **Medo (Fear):** 49.649 amostras (11.7%)
- **Tristeza (Sadness):** 121.187 amostras (28.7%)
- **Love (Amor):** 34.554 amostras (8.2%)
- **Raiva (Anger):** 59.317 amostras (14.1%)
- **Surpresa (Surprise):** 14.972 amostras (3.5%)

Este subconjunto permite abordar o problema como uma tarefa de classificação multiclasse, sendo particularmente útil para modelos que visam detetar emoções complexas no texto.

Devido aos recursos computacionais limitados e à elevada dimensão do conjunto de dados, optou-se por utilizar apenas cerca de 30% do total de frases disponíveis (mantendo as proporções entre classes), o que resultou num subconjunto de aproximadamente 126.824 frases.

Mesmo assim, esta amostra representa um volume substancial de dados, suficiente para treinar modelos eficazes de classificação emocional, mantendo uma boa representatividade entre as classes e garantindo a viabilidade do processamento dentro das restrições técnicas existentes.

- **Alegria (Joy):** 42.920 amostras (33.8%)
- **Medo (Fear):** 14.895 amostras (11.7%)
- **Tristeza (Sadness):** 36.356 amostras (28.7%)
- **Amor (Love):** 10.366 amostras (8.2%)
- **Raiva (Anger):** 17.795 amostras (14.1%)
- **Surpresa (Surprise):** 4.492 amostras (3.5%)

1.3 Classificação Binária vs. Classificação Multiclasse

A classificação é uma tarefa fundamental na aprendizagem automática, onde o objetivo é atribuir uma categoria ou classe a uma amostra com base nos seus atributos. No contexto

da análise de sentimentos e emoções, duas abordagens principais são utilizadas: a classificação binária e a classificação multiclasse.

Esta subsecção define ambos os conceitos e discute as suas diferenças, com ênfase nas implicações para o projeto em desenvolvimento.

1.3.1 Classificação Binária

A classificação binária refere-se a problemas onde os dados são categorizados em exatamente duas classes mutuamente exclusivas. Este tipo de classificação é adequado para tarefas que requerem uma distinção clara entre duas polaridades, como a análise de *feedback* de utilizadores.

Matematicamente, a classificação binária modela a probabilidade de uma amostra pertencer a uma das duas classes, frequentemente utilizando funções como a *sigmoide* em modelos como Regressão Logística. A métrica de avaliação típica com possibilidade de ajustar limiares de decisão para lidar com desequilíbrios de classes inclui:

- *Accuracy*: mede a proporção de previsões corretas em relação ao total de amostras. É útil quando as classes estão balanceadas.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- *Precision*: indica a proporção de amostras corretamente classificadas como positivas em relação a todas as classificadas como positivas. É importante quando os falsos positivos têm um custo elevado.

$$Precision = \frac{TP}{TP + FP}$$

- *Recall* (Sensibilidade): mede a proporção de positivos que foram corretamente identificados pelo modelo. É crucial em contextos onde falsos negativos são indesejáveis.

$$Recall = \frac{TP}{TP + FN}$$

- *F1-score*: é a média harmónica entre precisão e recall, oferecendo um equilíbrio entre ambas. É particularmente útil quando há um desbalanceamento entre as classes.

$$F1\text{-score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

onde TP, TN, FP, FN representam as quantidades de Positivos, Negativos, Falsos Positivos e Falsos Negativos, respectivamente.

A base de dados de sentimentos em estudo trata-se de um problema de classificação binária, visto que o objetivo é distinguir o sentimento das frases em 2 opções distintas: positivo ou negativo.

1.3.2 Classificação Multiclasse

A classificação multiclasse envolve a categorização de amostras em três ou mais classes. Esta abordagem é mais complexa do que problemas de classificação binária, pois requer que o modelo distinga nuances subtis para ambas as classes.

Matematicamente, estes modelos utilizam funções como *softmax* para estimar probabilidades nas várias classes, e a sua avaliação pode ser mais desafiante caso haja desequilíbrios entre classes ou confusões entre categorias semanticamente próximas.

A base de dados das emoções é um exemplo de um problema de classificação com 6 classes.

1.3.3 Principais Diferenças

As principais diferenças entre estas abordagens residem na complexidade do problema e nos requisitos computacionais.

A classificação binária é geralmente mais simples, exigindo menos dados e recursos computacionais, mas pode ser limitada em contextos onde várias categorias são relevantes.

Por outro lado, a classificação multiclasse, embora mais rica em informação, aumenta a dimensionalidade do problema e requer *datasets* maiores e mais diversificados para evitar *overfitting*. Além disso, a escolha do modelo e das técnicas de pré-processamento pode variar: por exemplo, *embeddings* como *Word2Vec* ou *GloVe* são particularmente úteis em multiclasse para captar relações semânticas entre emoções.

Capítulo 2

Pré-Processamento & Feature Engineering

2.1 Objetivos

Durante as semanas 4 a 6, o foco do projeto esteve na otimização do pré-processamento dos dados e na engenharia de atributos, preparando os textos para serem utilizados nos futuros modelos de aprendizagem automática. O principal objetivo foi refinar a limpeza dos dados, extrair representações numéricas adequadas e selecionar os atributos mais relevantes para melhorar o desempenho dos modelos.

Para definir os passos de pré-processamento, foi realizada uma extensa análise de vários atributos da base de dados, permitindo a adaptação das técnicas às particularidades do conjunto de dados utilizado.

Na quarta semana, foram implementadas técnicas avançadas de pré-processamento de texto, incluindo *stemming* e *lemmatization*, tratamento de contrações e negações e diferentes abordagens de tokenização. Estas melhorias, orientadas pelos resultados da análise dos atributos dos dados, permitiram uma representação mais consistente dos textos, ajustando o *pipeline* de pré-processamento às especificidades do conjunto de dados.

A quinta semana focou-se na conversão do texto em representações numéricas, explorando diferentes técnicas como *Bag of Words* (*BoW*), *TF-IDF* (*Term Frequency-Inverse Document Frequency*) e *N-grams*. Foram realizadas experiências com diferentes parâmetros para cada abordagem, avaliando o impacto das escolhas na qualidade das representações textuais com base nos atributos identificadas na análise inicial.

Por fim, na sexta semana, foram exploradas técnicas mais avançadas de engenharia de atributos, como *embeddings* de palavras (*Word2Vec*, *GloVe*, *FastText*), e técnicas de seleção de atributos, incluindo testes estatísticos como Qui-quadrado (χ^2), Informação Mútua e métodos baseados em modelos, como *Random Forest*. Estas abordagens permitiram identificar os atributos mais discriminantes, reduzindo a dimensionalidade e otimizando o desempenho dos modelos subsequentes.

Este capítulo detalha cada uma destas etapas, apresentando os métodos utilizados, os desafios enfrentados e os resultados obtidos.

2.2 Configuração do Ambiente de Desenvolvimento

A configuração do ambiente de desenvolvimento, realizada na primeira semana do projeto, foi essencial para garantir um fluxo de trabalho eficiente e colaborativo. Optou-se pela linguagem **Python** devido à sua versatilidade e suporte a bibliotecas específicas para Processamento de Linguagem Natural e aprendizagem automática. Para além disso, foi escolhido o **Jupyter Notebook** como ambiente de desenvolvimento principal, pela sua estrutura interativa que facilita a visualização de resultados, a experimentação incremental de código e a documentação

integrada, tornando-o ideal para análise exploratória e desenvolvimento de modelos.

As principais bibliotecas utilizadas incluíram NLTK para pré-processamento de texto (tokenização, *stemming*, lematização), Scikit-learn para algoritmos de *machine learning* e avaliação de modelos, Pandas e NumPy para manipulação de dados, BeautifulSoup para remoção de etiquetas *HTML*, Pyspellchecker para correção ortográfica, Contractions para expansão de contrações, Matplotlib e Seaborn para visualizações, Gensim para *embeddings* (Word2Vec, GloVe, FastText) e RE para limpeza de texto.

Para promover a colaboração entre os membros da equipa, foi também criado um repositório no *GitHub*, permitindo a partilha de ficheiros, rastreamento de alterações e revisão de código de forma eficiente.

2.3 Pré-Processamento dos Dados

Para garantir a qualidade dos dados e prepará-los para as etapas seguintes de análise e futura construção de modelos, foi implementada uma *pipeline* de pré-processamento de texto que combina técnicas básicas e avançadas de limpeza e normalização.

Esta secção descreve detalhadamente as etapas realizadas, os métodos utilizados e os resultados obtidos, com base na análise exploratória inicial do conjunto de dados e nas particularidades identificadas.

Esta *pipeline* foi desenhada para reduzir o ruído textual, padronizar o vocabulário e preservar informações relevantes para a análise de sentimentos, sendo aplicada e personalizada aos desafios de ambas as bases de dados em estudo.

2.3.1 Técnicas Básicas de Pré-Processamento

A *pipeline* de pré-processamento foi composta por uma sequência de etapas, aplicadas numa ordem específica para garantir os melhores resultados de acordo com o procurado.

Ao longo das subsecções seguintes, cada etapa será descrita em detalhe, com exemplos ilustrativos retirados dos conjuntos de dados.

Correção Ortográfica

A correção ortográfica é crucial na análise de sentimentos, pois elimina erros de escrita que podem comprometer a interpretação do texto pelos modelos de aprendizagem automática.

No contexto do projeto, que utiliza o *dataset* “Sentiment and Emotion Analysis” com frases de avaliações de produtos e redes sociais, palavras mal escritas (por exemplo, “fone” em vez de “phone”) podem ser interpretadas como termos distintos, aumentando a variabilidade do vocabulário e reduzindo a precisão da classificação de sentimentos. A correção ortográfica padroniza o texto, melhora a consistência lexical e facilita a extração de atributos semânticos, garantindo que o modelo se foque no conteúdo emocional em vez de ruídos ortográficos.

Esta correção foi implementada com o auxílio da biblioteca *pyspellchecker*, utilizando um dicionário personalizado que incorpora termos específicos do domínio, como “mic”, “phone”, “film” e “movie”. Palavras com menos de quatro caracteres ou já presentes no dicionário foram mantidas inalteradas para prevenir correções indevidas.

Por exemplo, a frase “*The fone is grate*” foi corrigida para “*The phone is great*”.

Expansão de Contrações e Possessivos

As contrações são formas reduzidas de palavras ou frases, criadas pela combinação de duas ou mais palavras com a omissão de letras, geralmente substituídas por apóstrofos, como “*don’t*” (*do not*), “*can’t*” (*cannot*) e “*it’s*” (*it is*). No (NLP), o tratamento de contrações é essencial para a

limpeza e normalização do texto, garantindo uniformidade e facilitando etapas subsequentes de análise. Nesta etapa, contrações e possessivos (e.g., “*don’t*” → “*do not*”, “*phone’s*” → “*phone*”) foram expandidos utilizando a biblioteca `contractions`.

Por exemplo, a frase “*It’s not working*” foi transformada em “*It is not working*”, promovendo maior consistência textual.

Conversão para Letras Minúsculas

A conversão de todo o texto para letras minúsculas padroniza a representação lexical, eliminando variações causadas por diferenças de capitalização (“*Bom*” vs. “*bom*”). Esta técnica foi aplicada utilizando a função `lower()`, garantindo uniformidade antes da tokenização.

Por exemplo, “*The Movie IS Great*” foi transformado em “*the movie is great*”. A conversão reduziu a variabilidade do vocabulário, facilitando a análise semântica e a compatibilidade com dicionários lexicais.

Remoção de Etiquetas HTML

Contendo textos provenientes de avaliações *online*, a base de dados apresentava etiquetas *HTML* residuais. Estas foram retiradas com o auxílio da biblioteca `BeautifulSoup`, que extrai apenas conteúdo lexical, eliminando ruído não semântico.

Remoção de Números

Por raramente contribuírem para a análise de sentimentos (exceto em contextos muitos específicos), números foram removidos, com o auxílio da biblioteca `re`. Esta técnica simplificou o vocabulário, reduzindo a dispersão lexical sem afetar o significado emocional das frases.

Remoção de Caracteres Especiais

A remoção de caracteres especiais (ex.: !, @, \$, %, &, *, etc.) eliminou símbolos que não contribuem para a análise semântica de sentimentos, reduzindo o ruído textual. Usando a biblioteca `re`, caracteres não alfanuméricos foram substituídos por espaços simples:

- “*Great product!!@#5*” → “*Great product*”
- “*Amazing!!!Movie...*” → “*Amazing Movie*”

Remoção de Espaços Extras

Espaços extras foram eliminados utilizando a função `strip()` e expressões regulares (`re.sub(r'\s+', ' ', text)`).

Por exemplo, “*This is great*” foi normalizado para “*This is great*”.

Remoção de Stopwords

A remoção de *stopwords* eliminou palavras comuns de baixo valor semântico, como preposições e conjunções (como, por exemplo, “*the*”, “*is*”, “*and*”), que raramente contribuem para a análise de sentimentos. A biblioteca `NLTK` forneceu uma lista padrão de *stopwords* em inglês, que foi adaptada para preservar termos cruciais, como “*not*”, “*never*” e “*no*”, devido à sua relevância em contextos de negação. Por fim, cada palavra tokenizada (palavra dividida em unidades menos chamados de *tokens*) foi comparada com a lista ajustada, sendo removida se considerada irrelevante.

Por exemplo, a frase “*This is a great movie*” foi transformada em “*great movie*”, mantendo apenas os termos com carga emocional. Esta etapa foi aplicada após a normalização de sinônimos e o tratamento de negações, garantindo que palavras contextuais fossem preservadas.

2.3.2 Técnicas Avançadas de Pré-Processamento

As técnicas avançadas de pré-processamento distinguem-se das abordagens básicas descritas anteriormente por operarem em níveis semânticos e sintáticos mais complexos.

Enquanto que as técnicas básicas se focam na limpeza e uniformização do texto, as avançadas visam preservar ou realçar o significado emocional e contextual, essencial para a análise de sentimentos e emoções.

2.3.2.1 Stemming vs. Lemmatization

Stemming e *lemmatization* são duas técnicas de normalização de texto que reduzem as palavras às suas formas base, simplificando a análise linguística e reduzindo a variabilidade do vocabulário da base de dados. A principal diferença entre elas está no nível de refinamento de cada técnica e na precisão com que processam os termos.

O *stemming* é uma abordagem mais rápida e direta, que remove afixos (*prefixos* e *sufixos*) para chegar a uma raiz aproximada, muitas vezes sem considerar se essa raiz é uma palavra válida no dicionário.

Por exemplo, a palavra “*running*” pode ser reduzida a “*run*”, mas, em alguns casos, palavras como “*university*” e “*universe*” podem acabar por ser reduzidas à mesma raiz, apesar de terem significados distintos. Esta técnica torna-se útil em processamentos de larga escala, onde a velocidade é prioritária, havendo sempre a possibilidade de gerar imprecisões.

Já a *Lemmatization* é uma técnica mais sofisticada, que recorre a regras linguísticas e dicionários para converter palavras às suas formas canónicas (*lemmas*), tendo em conta o seu contexto a classe gramatical.

Por exemplo, “*better*” é reduzido a “*good*”, e “*running*” mantém-se como “*run*”, mantendo os significados originais. Este método garante maior fidelidade ao significado original, sendo especialmente importante em problemas como aquele em estudo, que exigem uma interpretação semântica precisa.

A escolha da técnica de normalização mais adequada para o problema em questão será abordada na **Secção 2.3.3**.

2.3.2.2 Word Tokenization vs. Sentence Tokenization

A tokenização representa uma etapa fundamental no processamento de linguagem natural, envolvendo a divisão do texto em unidades menores conhecidas como *tokens*, feitas para facilitar a análise.

Existem duas abordagens principais para este processo: *Sentence Tokenization* (por frases) e *Word Tokenization* (por palavras).

A *sentence tokenization* segmenta o texto em unidades completas de significado (frases), mantendo intacta as suas estruturas sintáticas. Esta abordagem é especialmente útil para aplicações como tradução automática e resumos de textos, onde a preservação da estrutura frásica é crucial para manter o significado original. Vejamos o seguinte exemplo:

“O gato caiu da janela. Ele está bem, mas assustado.”

$$\text{Frase original} \rightarrow \begin{cases} 1. \text{ “O gato caiu da janela.”} \\ 2. \text{ “Ele está bem, mas assustado.”} \end{cases}$$

Por outro lado, a *word tokenization* divide o texto em unidades lexicais individuais, permitindo uma análise mais detalhada do conteúdo. Esta abordagem mostra-se particularmente eficaz para problemas como a classificação de sentimentos, onde a identificação de termos específicos e a sua carga emocional e subjetiva é essencial para atingir resultados confiáveis. Vamos agora ver a diferença deste método com o mesmo exemplo anterior:

“O gato caiu da janela. Ele está bem, mas assustado.”

Frase original → [“O”, “gato”, “caiu”, “da”, “janela”, “.”, “Ele”, “está” “bem”, “,”, “mas”, “assustado” .]

Considerando os objetivos deste trabalho, a tokenização por palavras foi selecionada como abordagem principal devido à sua capacidade de captar nuances emocionais presentes em termos isolados.

2.3.2.3 Tratamento de Negações

A negação pode alterar drasticamente o significado de uma frase, tornando essencial seu tratamento no processamento de linguagem natural. A presença de negações, como “*not happy*” ou “*not bad*”, pode mudar completamente o tom e sentimento da frase, afetando a interpretação pelos modelos de *NLP*. Se não forem devidamente tratadas, estas podem gerar erros na análise semântica e comprometer tarefas como tradução automática e classificação de sentimentos ou emoções.

Entre as abordagens para lidar com negações, destacam-se a substituição por antónimos, a marcação de palavras afetadas e a simplificação estrutural das frases.

Para abordar este desafio, foi implementada uma abordagem híbrida que combina a identificação de negações com a substituição por antónimos e a marcação de palavras afetadas. A biblioteca spaCy foi utilizada para análise de dependências sintáticas, permitindo identificar palavras negadas com maior precisão. A função `get_antonym` foi utilizada para substituir palavras negadas por antónimos sempre que possível, utilizando o recurso WordNet da biblioteca NLTK. Quando não existia um antónimo adequado, as palavras afetadas pela negação eram marcadas com o prefixo “NOT_”, preservando a informação contextual para os modelos de classificação.

Frase original: “I am not happy with this product.”

Processamento:

1. Identificação da negação:

“not happy” → Detetada negação no sintagma verbal

2. Substituição por antónimo (via WordNet):

“happy” → Antônimo: “unhappy”

3. Frase transformada:

Original → “I am not happy with this product.”

↓

Processada → “I am unhappy with this product.”

Caso sem antônimo direto: “This is not a good solution.”

1. Marcação contextual:

“not good” → “NOT_good” (preservação do contexto)

2. Frase final:

Original → “This is not a good solution.”

↓

Processada → “This is a NOT_good solution.”

2.3.2.4 Normalização de Sinónimos

A normalização de sinónimos ajuda a reduzir a variabilidade lexical do *dataset*, agrupando palavras semanticamente semelhantes numa forma canónica para minimizar a dispersão do vocabulário.

Com base na análise exploratória, que identificou termos emocionais frequentes (como, por exemplo: “great”, “awesome”, “fantastic” nas avaliações positivas), foi construído um dicionário manual mapeando esses termos para formas padronizadas (ex.: “great”, “awesome”, “fantastic” → “good”; “horrible”, “awful”, “terrible” → “bad”). Este dicionário foi complementado por consultas dinâmicas ao WordNet da biblioteca NLTK, que selecionou sinónimos adicionais considerando a classe gramatical (*POS*) e o contexto emocional.

Com esta técnica, a frase “*The film was awesome*” foi transformada em “*The film was good*”, e “*This product is horrible*” em “*This product is bad*”.

2.3.3 Pipeline Final de Pré-Processamento para ambos os Datasets

Considerando os requisitos específicos de ambas as bases de dados e futuros modelos de classificação a ser desenvolvidos, foi definida uma *pipeline* final de pré-processamento que integra técnicas otimizadas para reduzir ruído textual, preservar significado semântico e emocional e preparar os dados para a criação de modelos.

Este *pipeline* foi desenhado para ser aplicado a ambas as bases de dados, com ajustes mínimos para acomodar as particularidades de cada um (destaca-se, por exemplo, a preservação de *stopwords* emocionais no *dataset* das emoções).

2.3.3.1 Ordem da Pipeline Final

A *pipeline* final segue a sequência de etapas conforme descrito abaixo:

1. Correção de Erros Ortográficos
2. Expansão de Contrações e Possessivos
3. Tratamento de Negações
4. Normalização de Sinónimos
5. Conversão para Letras Minúsculas
6. Remoção de Etiquetas HTML
7. Remoção de Números
8. Remoção de Caracteres Especiais
9. Remoção de Espaços Extras
10. Remoção de Stopwords

```
def create_default_preprocessor():
    """Create a preprocessor with default steps in a specific order"""
    preprocessor = TextPreprocessor()
    preprocessor.add_step(correct_spelling)
    preprocessor.add_step(expand_contractions_and_positives)
    preprocessor.add_step(handle_negation)
    preprocessor.add_step(normalize_synonyms)
    preprocessor.add_step(to_lowercase)
    preprocessor.add_step(remove_html_tags)
    preprocessor.add_step(remove_numbers)
    preprocessor.add_step(remove_special_chars)
    preprocessor.add_step(remove_extra_spaces)
    preprocessor.add_step(remove_stopwords,
                         stop_words=get_stopwords(keep_contextual=True))
    return preprocessor
```

Figura 2.1: Pipeline final de pré-processamento e sua implementação em Python.

Esta ordem de etapas foi escolhida propositadamente, de maneira a realizar transformações contextuais (negações e sinônimos, por exemplo) antes da remoção de informações (remoção de *stopwords* e caracteres especiais, por exemplo).

Por fim, a aplicação de *lemmatization* e *stemming* foi realizada após a criação da *pipeline* original, de maneira a avaliar qual dos processamentos obtinha melhores resultados.

2.3.3.2 Resultados Finais do Pré-Processamento

A avaliação do impacto da *pipeline* foi realizada com a função `evaluate_preprocessing`, que calcula métricas como redução de tamanho, redução de vocabulário, sobreposição lexical entre classes e preservação da distribuição de classes. Os resultados para ambos os *datasets* são apresentados na Tabela 2.1.

Tabela 2.1: Métricas de avaliação do impacto do pré-processamento nos *datasets* de sentimentos e emoções.

Métrica	Sentimentos	Emoções
Redução média de tamanho	42.2%	46.0%
Redução de vocabulário	8.6%	0.5%
Sobreposição lexical entre classes	27.8%	33.0%
Preservação da distribuição de classes	100%	100%

Para além disso, foi utilizado um classificador *Multinomial Naive-Bayes* com *Countvectorizer* (*n-gramas* de 1 a 2) para avaliar o desempenho de diferentes variantes do texto processado.

As avaliações realizadas anteriormente permitiram avaliar a eficiência do melhor método de pré-processamento para cada base de dados, consoante as seus atributos, podendo também analisar as diferenças e semelhanças entre a análise de sentimentos e emoções.

Foi-nos permitido tirar as seguintes conclusões:

- **Redução de Tamanho:** O *dataset* de emoções apresentou maior redução média de tamanho (46.0%) em comparação com o de sentimentos (42.2%), devido à maior prevalência de *stopwords* e caracteres especiais no texto emocional, que foram eficientemente removidos. Esta redução indica uma limpeza significativa de ruído textual em ambos os casos.
- **Redução de Vocabulário:** A redução de vocabulário foi mais pronunciada no *dataset* de sentimentos (8.6%) do que no de emoções (0.5%), refletindo uma maior diversidade lexical no *dataset* de emoções, com 422.746 frases e seis classes, que dificultou a consolidação lexical via normalização de sinônimos. No entanto, a baixa redução no *dataset* de emoções sugere que a *pipeline* preservou a riqueza lexical necessária para distinguir as diferentes classes.
- **Sobreposição Lexical:** A sobreposição lexical entre classes foi maior no *dataset* de emoções (33.0%) do que no de sentimentos (27.8%), o que é esperado devido ao maior número de classes (seis vs. duas) e à presença de palavras comuns como “feel” e “feeling”. Apesar disso, a sobreposição moderada em ambos indica que a *pipeline* distinguia diferenças lexicais entre classes.
- **Preservação da Distribuição:** Ambos os *datasets* mantiveram a distribuição original de classes, confirmando que a *pipeline* não introduziu perdas ou desbalanceamento.
- **Desempenho do Modelo:** No *dataset* de sentimentos, a variante processada alcançou a maior *accuracy* (0.907) e *F1-score* (0.909), superando o texto original (0.888 e 0.892). A lematização ofereceu ganhos marginais no *recall* (0.915).

Já no *dataset* de emoções, a variante processada obteve o melhor desempenho (*accuracy* de 0.821, *F1-score* de 0.802), significativamente superior ao texto original (0.770 e 0.740). Lematização e *stemming* apresentaram ganhos menores em emoções, devido à complexidade multiclasse.

Para além disso, a análise de palavras frequentes no *dataset* de emoções revelou que termos como “*feel*” (com 31.248 ocorrências na classe “*joy*”) e “*feeling*” dominam todas as classes, refletindo a natureza introspectiva do texto. A preservação de *stopwords* contextuais e intensificadores (por exemplo, “*really*”) foi crucial para captar nuances emocionais, contribuindo para o desempenho superior da variante processada.

Conforme esta análise, na busca e construção dos melhores modelos para cada *dataset*, foram testados tanto o texto processado, como o texto processado e lematizado.

2.4 Engenharia de Atributos

A engenharia de atributos constitui uma etapa fundamental no desenvolvimento de modelos de Processamento de Linguagem Natural, servindo como ponte entre os dados textuais pré-processados e os algoritmos de *machine learning*.

No contexto deste projeto, a escolha das técnicas de representação textual foi orientada pelas especificidades das bases de dados de sentimentos e emoções, que apresentam desafios distintos, como a necessidade de captar nuances emocionais subtis e lidar com vocabulários variados. As abordagens selecionadas incluem métodos baseados em frequência, como *Bag of Words* (*BoW*), *N-gramas* e *TF-IDF*, que oferecem interpretabilidade e simplicidade, e métodos baseados em *embeddings*, como *Word2Vec*, *GloVe* e *FastText*, que capturam relações semânticas mais profundas.

Nesta seção, será abordado o funcionamento teórico dessas técnicas, detalhando como elas transformam textos dos *datasets* em formatos adequados para modelos de classificação.

2.4.1 Técnicas de Representação de Texto

A representação textual é a base para a análise de sentimentos e emoções, transformando dados não estruturados, como frases e parágrafos, em estruturas numéricas que algoritmos de *machine learning* podem interpretar.

Métodos baseados em frequência, como *BoW* e *TF-IDF* oferecem uma representação interpretável que preserva a informação sobre a ocorrência de termos-chave, particularmente relevante para a análise de sentimentos e emoções.

2.4.1.1 Bag of Words (BoW)

A técnica *Bag of Words* converte textos em vetores numéricos com base na contagem de palavras, ignorando a sua ordem ou estrutura gramatical.

Para o *dataset* de sentimentos, uma frase como “*The movie is great*” é dividida em *tokens* (“*The*”, “*movie*”, “*is*”, “*great*”), e um vocabulário único é criado com todas as palavras do *dataset*. Cada frase é representada como um vetor, onde cada posição indica a frequência de uma palavra do vocabulário (quantas vezes esta aparece).

Por exemplo, “*great*” pode ter um valor alto em frases positivas, enquanto que palavras ausentes recebem zero. No *dataset* de emoções, uma frase como “*I feel overwhelming joy*” seria processada da mesma forma, com *tokens* como “*joy*” seriam destacados para a classe *Joy*.

A simplicidade do *BoW* permite captar palavras-chave emocionais, como “*awesome*” no *dataset* de sentimentos ou “*fearful*” no *dataset* de emoções, mas a sua limitação está na incapacidade de captar contexto, como a diferença entre “*I am happy*” e “*I am not happy*”. Esta

abordagem é computacionalmente leve e interpretável, sendo, no entanto, menos eficaz para nuances emocionais complexas no *dataset* de emoções.

Figura 2.2: Exemplo *BoW*



BoW Model Evaluation:
BoW - Accuracy: 0.831

Figura 2.3: *BoW Model Evaluation Accuracy* utilizando *Naive Bayes*

2.4.1.2 N-grams

A técnica de *N-grams* pode considerar-se como uma extensão do *BoW*, ao considerar sequências contínuas de n palavras, onde n é um número inteiro positivo que define o tamanho da sequência, permitindo a captura de contexto linguístico local.

Ao contrário do *BoW*, que trata palavras como unidades isoladas ignorando a ordem, os *N-grams* preservam a relação entre palavras adjacentes, modelando padrões sintáticos e semânticos. Estas sequências de palavras ou *tokens* são extraídas do texto após a tokenização, onde cada *token* pode ser uma palavra, pontuação ou outro elemento linguístico. O valor de n determina o grau de contexto capturado: um n pequeno (1 ou 2) irá focar em relações locais, enquanto que um n maior (3 ou mais) capta sequências mais longas, como expressões idiomáticas ou frases completas. Cada *N-gram* é tratado como uma unidade única, e o texto é representado como um vetor que contabiliza a frequência dessas sequências no vocabulário do *corpus*.

A principal vantagem dos *N-grams* é a sua capacidade de captar contexto local, enriquecendo a representação textual em comparação com o *BoW*. No entanto, o aumento do n aumenta a dimensionalidade do vocabulário, pois o número de *N-grams* possíveis cresce exponencialmente, aumentando por sua vez o custo computacional e o risco de *overfitting*, especialmente em *datasets* com vocabulários diversos. Além disso, *N-grams* de ordem superior podem captar padrões mais complexos, mas sequências muito longas podem ser esparsas, aparecendo em poucas frases, o que reduz sua utilidade estatística.

No contexto deste projeto, foram realizados maioritariamente estudos com *1-grams*, *2-grams* e *3-grams*, considerando o equilíbrio entre captação de contexto e viabilidade computacional.

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

O TF-IDF é uma das técnicas mais fundamentais em *Information Retrieval* e Processamento de Linguagem Natural. Esta medida estatística avalia a importância de um termo dentro de um documento relativo a uma coleção de documentos (corpus), combinando dois fatores essenciais:

$$TF-IDF(t, d, D) = \underbrace{TF(t, d)}_{\substack{\text{Frequência do termo} \\ \text{no documento}}} \times \underbrace{IDF(t, D)}_{\substack{\text{Inverso da frequência} \\ \text{no corpus}}} \quad (2.1)$$

Componentes do TF-IDF:

1. *Term Frequency* (TF)

Calcula a frequência normalizada de um termo t no documento d :

$$TF(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (2.2)$$

onde:

- $f_{t,d}$ = número de ocorrências do termo t no documento d
- $\sum_{t' \in d} f_{t',d}$ = soma das frequências de todos os termos em d

2. *Inverse Document Frequency* (IDF)

Mede a importância geral do termo no corpus:

$$IDF(t, D) = \log \left(1 + \frac{N}{n_t} \right) \quad (2.3)$$

onde:

- N = número total de documentos no corpus D
- n_t = número de documentos contendo o termo t

3. Cálculo Final do TF-IDF

O valor final combina ambos os componentes:

$$TF-IDF(t, d, D) = \left(\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \right) \times \log \left(1 + \frac{N}{n_t} \right) \quad (2.4)$$

Exemplo Numérico:

Consideremos:

- Corpus com $N = 10,000$ documentos
- Termo "data" aparece 25 vezes em um documento com 500 palavras
- Termo "data" aparece em 1,000 documentos do corpus

$$TF("data", d) = \frac{25}{500} = 0.05$$

$$IDF("data", D) = \log \left(1 + \frac{10000}{1000} \right) \approx 2.30$$

$$TF-IDF("data", d, D) = 0.05 \times 2.30 \approx 0.115$$

Interpretação:

- **TF alto + IDF alto:** Termo importante e específico (ex: termos técnicos)
- **TF alto + IDF baixo:** Termo comum (ex: stopwords)
- **TF baixo + IDF alto:** Termo raro mas potencialmente relevante

2.4.2 Word Embeddings

Os *Word Embeddings* representam uma evolução significativa no campo da representação textual, superando muitas das limitações das abordagens tradicionais baseadas em frequência.

Estas representações vetoriais densas permitem captar não apenas a presença de termos isolados, mas também relações semânticas complexas entre palavras, incluindo similaridades, analogias e gradientes com significados particularmente relevantes para a análise emocional.

No contexto do nosso trabalho, implementámos e comparámos duas abordagens principais: *embeddings* pré-treinados, que incorporam conhecimento linguístico geral, e *embeddings* específicos do domínio, treinados diretamente nos nossos dados para captar as particularidades do vocabulário emocional presente nos nossos *datasets*.

Esta análise permitiu-nos identificar o equilíbrio óptimo entre generalização e especialização para cada uma das bases de dados.

2.4.2.1 Word2Vec

Word2Vec é uma técnica baseada em redes neurais que aprendem representações vetoriais de palavras com base nos seus contextos. Esta opera sob o princípio de que palavras que aparecem em contextos semelhantes devem ter vetores próximos no espaço vetorial. Existem dois principais modelos no Word2Vec:

- **CBOW (Continuous Bag of Words):** Este modelo prevê uma palavra-alvo com base nas palavras que a rodeiam (ou seja, o seu contexto). Por exemplo, dado o contexto {gato, está, cima, da}, o modelo tenta prever a palavra central em. Matematicamente, este maximiza a probabilidade $P(w_t|w_{t-k}, \dots, w_{t+k})$, onde w_t é a palavra-alvo e w_{t-k}, \dots, w_{t+k} são as palavras de contexto num intervalo k .
- **Skip-gram:** Este modelo faz o oposto: dada uma palavra-alvo, tenta prever as palavras do seu contexto. Por exemplo, dada a palavra em, o modelo tenta prever palavras como gato, está, cima, da. Matematicamente, este maximiza a probabilidade $P(w_{t+j}|w_t)$ para cada palavra de contexto w_{t+j} .

No dataset de sentimentos, o *Skip-gram* é teoricamente vantajoso, pois enfatiza palavras-chave emocionais, como “awesome” em “The movie is awesome”, associando-a a termos como “great” ou “excellent”. Como o dataset de sentimentos tem um vocabulário limitado, o *Skip-gram* pode captar relações semânticas precisas com menos dados. Por outro lado, no dataset de emoções, o *CBOW* é mais adequado, pois lida com um volume de dados significativamente maior, prevendo palavras como “sadness” a partir de contextos como “loss” ou “tears”. A capacidade

do Word2Vec de modelar similaridades semânticas, como “happy” próximo de “joyful”, é crucial para distinguir classes como Joy e Love no dataset de emoções. No entanto, o treino desta técnica é computacionalmente intensivo, especialmente para o dataset de emoções, e depende de uma janela de contexto bem ajustada para evitar captar relações irrelevantes.

2.4.2.2 GloVe

O GloVe (*Global Vectors for Word Representation*) é um modelo de representação de palavras que combina estatísticas globais de cocorrência com aprendizagem contextual. Ao contrário do Word2Vec que processa contextos locais de forma sequencial, o GloVe constrói uma matriz de cocorrência X , onde X_{ij} representa a frequência com que as palavras w_i e w_j aparecem juntas numa determinada janela de contexto. O modelo otimiza a seguinte função objetivo:

$$J = \sum_{i,j=1}^V f(X_{ij})(w_i^T \bar{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \quad (2.5)$$

, onde

- w_i e \bar{w}_j são os vetores de palavras e vetores de contexto, respectivamente.
- b_i e \tilde{b}_j são termos de viés para cada palavra.
- V é o tamanho do vocabulário.
- $f(X_{ij})$ é uma função de ponderação que reduz o impacto de coocorrências raras.
- $\log(X_{ij})$ é o logaritmo da contagem de coocorrência.

A descida de gradiente otimiza os vetores para captar relações semânticas globais, aproximando o produto escalar dos vetores de palavras ao logaritmo da probabilidade de coocorrência.

A construção da matriz de coocorrência no modelo *GloVe* é fundamental para contabilizar a frequência de pares de palavras dentro de uma determinada janela de contexto. Por exemplo, num *dataset* de sentimentos, uma janela de 3 palavras pode captar relações como “great movie” na frase “This is a great movie”. A função $f(X_{ij})$ é utilizada para mitigar o impacto de coocorrências raras, sendo necessários ajustes em *datasets* de sentimentos para evitar superestimação devido ao seu tamanho reduzido. Os vetores finais do *GloVe* são obtidos combinando vetores de palavras e vetores de contexto, geralmente por meio de soma ou concatenação. Em *datasets* de sentimentos, a concatenação tende a destacar relações como “awesome” e “great”, enquanto que em *datasets* de emoções, a soma é preferível para reduzir a dimensionalidade sem perder informações relevantes para classes como “Love” e “Anger”.

O *GloVe* destaca-se como um modelo eficiente na captura de coocorrências globais, sendo escalável tanto para *datasets* extensos (como os de emoções) quanto para bases de dados menores (como os de sentimentos). A sua abordagem baseada em fatorização de matrizes permite modelar relações semânticas globais, como analogias do tipo “good” próximo de “great”, com menor dependência de *fine tuning* na janela de contexto. Além disso, o *GloVe* é computacionalmente eficiente para vocabulários amplos, beneficiando-se de uma construção rápida da matriz de coocorrência, especialmente em *datasets* bem estruturados. No entanto, apresenta limitações ao adaptar-se a vocabulários específicos, particularmente para classes raras, onde nuances contextuais podem ser menos capturadas.

Em comparação com o *Word2Vec*, que modela contextos locais por meio de redes neurais e é mais adaptável a vocabulários limitados, o *GloVe* mostra-se mais eficiente para grandes volumes de dados. Contudo, a sua dependência de estatísticas globais pode resultar em menor

precisão para nuances emocionais específicas, tornando o *Word2Vec* mais adequado em cenários onde a captura de detalhes contextuais é crítica. Assim, a escolha entre os dois modelos depende do contexto de aplicação, do tamanho do *dataset* e da necessidade de captar relações semânticas mais globais ou locais.

2.4.2.3 FastText

A técnica de *FastText*, extensão do *Word2Vec*, é uma ferramenta de aprendizagem supervisionada de linguagem natural que cria *embeddings* de palavras a partir de “subpalavras” (*n-grams*). Ao contrário de modelos como *Word2Vec* e *GloVe*, que tratam cada palavra como uma unidade individual, o *FastText* usa subpalavras para gerar *embeddings* de palavras, incluindo aquelas que não foram vistas durante o treino. Por exemplo, a palavra “happiness” é decomposta em *3-grams* como “*hap*”, “*pi*”, “*ness*”, com o seu vetor final obtido através da soma dos vetores destes *3-grams*.

Esta abordagem permite não só gerar representações para palavras fora do vocabulário, como também captar informações morfológicas, melhorando a generalização para palavras raras ou derivadas.

Este utiliza o modelo *Skip-gram* do *Word2Vec*, maximizando a probabilidade $P(w_{t+j}|w_t)$, onde w_t é representada pelos seus *n-grams*, incorporando subpalavras à função objetivo.

Na base de dados dos sentimentos, o *FastText* poderá ser vantajoso ao captar variações morfológicas. Já na base de dados das emoções, a sua robustez poderá ser particularmente valiosa para classes como “*Surprise*”, onde palavras raras ou derivadas como “*shockingly*” em “*I'm shockingly surprised*”, são frequentes, tornando-se ideal para modelar vocabulários variados e generalizar para termos novos.

Entre as suas vantagens, destaca-se a capacidade de lidar com palavras raras e morfologicamente complexas, além de sua adaptabilidade a *datasets* com diversidade lexical. Contudo, a modelagem de *n-grams* aumenta o custo computacional em comparação com o *GloVe*, devido à maior complexidade na representação de subpalavras.

Capítulo 3

Seleção de Modelos & Combinacões de Parâmetros

3.1 Objetivos

Durante a semana 7, o foco do projeto esteve na construção e treino dos modelos de classificação para análise de sentimentos. O objetivo foi selecionar modelos adequados, estabelecer uma linha de base de desempenho para, posteriormente, otimizar os modelos através do ajuste de hiperparâmetros para melhorar a precisão da classificação.

Na sétima semana, foram explorados diferentes algoritmos de classificação, sendo estes: *Random Forest*, Regressão Logística e *Support Vector Machines* (SVM). O conjunto de dados foi dividido em treino, validação e teste, e os modelos foram treinados com os atributos extraídos nas semanas anteriores. A avaliação inicial foi realizada utilizando métricas como *accuracy*, precisão, *recall* e *F1-score*, permitindo a documentação de um desempenho de referência para cada modelo.

Este capítulo apresenta os detalhes destas etapas, abordando a escolha dos modelos, a avaliação inicial e os resultados obtidos para as melhores combinações de parâmetros.

3.2 Seleção dos Modelos a ser Avaliados

A escolha dos modelos de classificação foi feita considerando a necessidade de lidar com dados textuais de alta dimensionalidade, esparsos e com nuances semânticas complexas. Os três algoritmos selecionados - *Random Forest*, Regressão Logística e *SVM* - oferecem diferentes abordagens para a classificação, complementando-se em termos de interpretabilidade, robustez e capacidade de generalização.

A seguir, apresentamos os princípios teóricos e atributos por detrás de cada modelo.

3.2.1 Random Forest

A *Random Forest* é um método de previsão baseado em árvores de decisão, que combina múltiplas árvores para realizar previsões mais robustas. Cada árvore é construída utilizando uma amostra aleatória com reposição dos dados de treino (*bagging*) e um subconjunto aleatório de atributos em cada nó, o que reduz a correlação entre as árvores e minimiza o risco de *overfitting*. A predição final é obtida pela votação maioritária (para classificação) das árvores individuais.

Matematicamente, para uma entrada x , a previsão deste método é dada por:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \dots, h_T(x)\}$$

onde $h_t(x)$ é a previsão da t -ésima árvore e T é o número total de árvores.

Este método é particularmente eficaz para lidar com a variabilidade das representações textuais (BoW ou TF-IDF). Para além disso, o *Random Forest* consegue lidar bem com atributos

de alta dimensionalidade, comuns em tarefas de análise de texto, e oferece interpretabilidade através da análise da importância dos atributos, facilitando a identificação de termos-chave relevantes para a classificação.

3.2.2 Regressão Logística

Apesar do nome, a Regressão Logística é um modelo de classificação que estima a probabilidade de uma amostra pertencer a uma classe específica, utilizando a função sigmoide.

Para a classificação binária, a probabilidade de uma amostra x pertencer à classe positiva é calculada como:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}},$$

onde β_i são os coeficientes aprendidos e x_i são os atributos.

Já para a classificação multiclasse, é utilizada a função softmax para estimar as probabilidades de cada classe. De seguida, o dataset é treinado minimizando a função de perda logarítmica (log-loss) com regularização (L1 ou L2) para evitar *overfitting*.

A simplicidade deste método torna-o uma escolha eficiente para problemas de classificação textual. Para além disso, a sua interpretabilidade, derivada dos coeficientes associados a cada atributo, e a sua eficiência computacional permitem análises rápidas, mesmo em cenários com datasets de dimensões moderadas.

3.2.3 Support Vector Machine (SVM)

O método SVM procura encontrar o hiperplano que maximiza a margem entre as classes, separando os pontos mais próximos (vetores de suporte). Para dados não linearmente separáveis, o SVM utiliza o “kernel trick”, como o kernel polinomial ou RBF, para mapear os dados num espaço de maior dimensão onde a separação é possível.

Matematicamente, o SVM resolve o problema de otimização:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i,$$

sujeito a $y_i(w^T x_i + b) \geq 1 - \xi_i$, onde w é o vetor normal ao hiperplano, b é o viés, C é o parâmetro de regularização e ξ_i são as variáveis de folga que permitem erros de classificação. A robustez do SVM a dados esparsos e de alta dimensionalidade torna-o ideal para a aplicação em conjunto com representações textuais, como TF-IDF ou *embeddings*.

A sua capacidade de lidar com ambiguidades, especialmente com o kernel RBF, permite captar nuances semânticas complexas, enquanto que o kernel linear é particularmente eficaz para dados textuais de alta dimensionalidade, como os encontrados em tarefas de classificação.

3.3 Escolha das Combinações de Atributos

A análise das várias combinações de atributos foi uma etapa fundamental para a seleção dos modelos mais promissores a serem desenvolvidos e otimizados. Estas combinações foram projetadas para explorar representações baseadas em frequência, semântica e contexto, com o objetivo de captar informações discriminantes para a classificação de sentimentos, e selecionar as abordagens mais promissoras para desenvolvimento e otimização futura.

O processo de análise envolveu a avaliação de cada combinação com três classificadores (Logistic Regression, Random Forest e SVM) e explorar as suas complementariedades para identificar as configurações mais adequadas para a classificação dos problemas em questão.

Nas subsecções seguintes, serão detalhadas as combinações testadas, o processo de análise e as adaptações realizadas para cada dataset, considerando tanto o texto processado (`processed_text`)

como o texto processado com lematização (`processed_lemmatized`). Todas as combinações foram geradas e testadas em ambos os formatos de texto, garantindo uma avaliação abrangente para a seleção de modelos.

3.3.1 Combinações Testadas para o Dataset de Sentimentos

A análise do dataset de sentimentos serviu como referência para a abordagem aplicada ao dataset de emoções. O processo começou com a divisão dos dados em 70% para treino, 15% para validação e 15% para teste, seguido por validação cruzada com 5 *folds* para garantir estimativas robustas de desempenho e apoiar a seleção de modelos.

As combinações de atributos foram projetadas para explorar representações baseadas em frequência, semântica e contexto, visando captar informações discriminantes para a classificação de sentimentos. A escolha das combinações considerou diversidade (representações de frequência e semântica), interpretabilidade (facilitando a identificação de palavras-chave) e viabilidade computacional (com restrições de `max_features=1000` e `k=300` para reduzir dimensionalidade).

Representações Individuais:

- **Bag of Words (BoW):** Representação baseada na contagem de palavras, configurada com `max_features=1000` e n-gramas: unigramas (`ngram_range=(1,1)`), bigramas (`ngram_range=(2,2)`) e trigramas (`ngram_range=(3,3)`).
Selecionada pela simplicidade e capacidade de captar termos relevantes.
- **TF-IDF:** Representação ponderada pela frequência de termos no documento e sua raridade no *corpus*, com os mesmos parâmetros de `max_features=1000` e n-gramas.
Incluída para destacar termos discriminantes, minimizando o impacto de palavras comuns.
- **Word2Vec:** Embeddings treinados no dataset com `vector_size=100`, utilizando o modelo Skip-gram (`sg=1`). Vetores de sentença foram gerados pela média dos vetores de palavras, capturando relações semânticas.
- **GloVe:** Embeddings pré-treinados com dimensão 100, com vetores de sentença obtidos pela média das palavras, incorporando conhecimento linguístico geral.
- **FastText:** Embeddings treinados no dataset com `vector_size=100`, aproveitando subpalavras para lidar com variações morfológicas e palavras raras.

Combinações de N-gramas:

Foram configuradas combinações de unigramas, bigramas e trigramas nas técnicas de BoW e TF-IDF, utilizando `ngram_range=(1,3)` para captar contexto local e preservar informações de sequências de palavras.

Combinações de Representações:

- **BoW+TF-IDF:** Concatenação de matrizes esparsas de BoW e TF-IDF, combinando frequência bruta com relevância ponderada.
- **BoW/TF-IDF+Embeddings:** Concatenação de representações esparsas (BoW ou TF-IDF) com embeddings densos (Word2Vec, GloVe ou FastText), explorando a complementaridade entre frequência e semântica.
- **Word2Vec+GloVe+FastText:** Combinação de embeddings para avaliar sinergias entre representações semânticas.

Combinações com Seleção de Atributos:

- **Chi-squared (χ^2):** Seleção de 300 atributos ($k=300$) com base no teste Qui-quadrado, aplicado a BoW e TF-IDF, identificando termos com maior associação às classes de sentimento.
- **Mutual Information (MI):** Seleção de $k=300$ atributos com base na informação mútua, priorizando termos com alta dependência às classes.
- **Random Forest (RF):** Seleção de $k=300$ atributos com base na importância calculada por um modelo Random Forest (`n_estimators=100`), destacando termos preditivos.

Alguns exemplos incluem BoW+ χ^2 , TF-IDF+Mutual Information e BoW+Word2Vec+Random Forest.

Todas estas combinações permitiram que fossem realizadas 48 avaliações distintas por modelo de classificação (Random Forest, Regressão Logística e SVM), resultando num total de 144 combinações testadas para cada formato de texto (processado e processado+lematizado).

3.3.2 Particularidades para o Dataset de Emoções

O dataset de emoções apresentou alguns desafios maioritariamente computacionais (como limitações de memória), que exigiram adaptações específicas para viabilizar a análise.

Para lidar com essas restrições, o dataset de análise foi reduzido para 40% do tamanho original, mantendo a proporção das classes, e a validação cruzada foi omitida, com o desempenho avaliado diretamente na divisão de teste (70% treino, 15% validação, 15% teste). Todas estas adaptações foram motivadas pela necessidade de reduzir a carga computacional e garantir que a análise permanecesse viável sem comprometer a captura de nuances emocionais.

Representações Individuais

- **BoW:** Configurada com `max_features=1000` e `ngram_range=(1, 2)` (unigramas e bigramas), priorizando simplicidade e eficiência computacional.
- **TF-IDF:** Configurada com os mesmos parâmetros de BoW, destacando termos relevantes para as classes emocionais.
- **Word2Vec:** Treinado com o modelo CBOW (`sg=0, vector_size=100`) para maior eficiência, com vetores de sentença gerados pela média das palavras.
- **GloVe:** Embeddings pré-treinados com dimensão 100, utilizando a média dos vetores de palavras.
- **FastText:** Treinado com `vector_size=100`, priorizado por sua capacidade de lidar com palavras raras e variações morfológicas, comuns em textos emocionais.

Combinações de N-gramas

Foram utilizadas combinações de unigramas e bigramas em BoW e TF-IDF (`ngram_range=(1, 2)`), considerando as limitações de memória referidas anteriormente.

Combinações de Representações

- **BoW + TF-IDF:** Concatenação das representações para combinar frequência e relevância.

- **TF-IDF + Embeddings:** Foram priorizadas combinações com FastText devido à sua robustez com palavras raras, especialmente relevantes para emoções.
- **Word2Vec + GloVe:** Testadas para explorar sinergias entre embeddings semânticos.

Combinações com Seleção de Atributos

- **Chi-squared (Chi^2):** Seleção de $k=300$ atributos aplicados a BoW e TF-IDF, para identificar termos com alta associação com as classes emocionais.
- **Mutual Information (MI):** Seleção de $k=300$ atributos para destacar termos com forte dependência com as classes.
- **Random Forest (RF):** Seleção limitada de $k=300$ atributos devido ao custo computacional, priorizando termos com maior impacto preditivo.

Exemplos incluem TF-IDF+ Chi^2 e BoW + FastText + Mutual Information.

Todas estas combinações permitiram que fossem realizadas 48 avaliações distintas por modelo de classificação (Random Forest, Regressão Logística e SVM), resultando num total de 144 combinações testadas para cada formato de texto (processado e processado+lematizado).

3.4 Avaliação e Combinações Finais

Após a seleção dos modelos e combinações a serem avaliados, foram desenvolvidos dois ficheiros Python: `combinationanalysis_sentiment.ipynb` e `combinationanalysis_emotion.ipynb`. O objetivo principal destes ficheiros foi analisar o desempenho de todas as combinações de atributos e identificar um top 3 para utilização nas etapas subsequentes do projeto.

3.4.1 Dataset de Sentimentos

3.4.1.1 Ranking de Modelos

De entre os 3 modelos escolhidos para esta avaliação (Random Forest, Regressão Logística e Random Forest), a **Regressão Logística** destacou-se como o modelo com os melhores desempenhos, alcançando a maior *accuracy* de 0.9320 na combinação TF-IDF_uni+bi. Este destaque justifica-se com a capacidade deste modelo de lidar com dados textuais de alta dimensionalidade e esparsos, tal como o como BoW e TF-IDF, aliado à capacidade de modelar relações lineares e à eficiência computacional.

O **SVM** apresentou um desempenho competitivo, porém inferior ao da RL, com *accuracies* próximas, como 0,9250 na mesma combinação TF-IDF_uni+bi, beneficiando do kernel linear, mas com um custo computacional mais elevado.

Por fim, o **Random Forest** registou o desempenho mais baixo entre os três, com *accuracies* inferiores (por exemplo, 0,9100 em TF-IDF_uni+bi), devido à sua menor adaptabilidade a dados esparsos de alta dimensionalidade.

3.4.1.2 Ranking de Processamento

Nesta base de dados, o texto processado (`processed_text`) alcançou resultados ligeiramente superiores, com *accuracy* média de 0,907 contra 0,900 para o texto lematizado. A preservação das formas originais das palavras parece captar nuances semânticas mais relevantes para a classificação binária de sentimentos, onde o vocabulário é menos variado.

3.4.1.3 Top 3 de Combinações

Segue-se abaixo uma visualização *heatmap*, com todas as combinações realizadas para os dados processados:

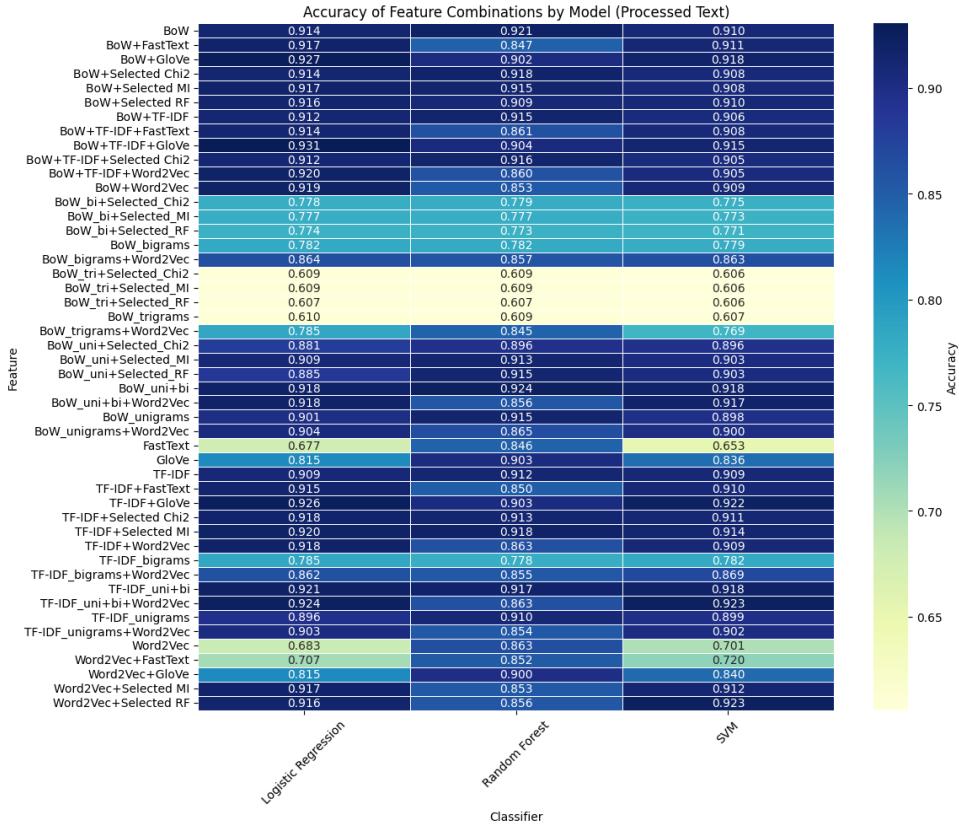


Figura 3.1: Combinações testadas para o texto processado e respetivos resultados

Finalmente, a Tabela 3.1 apresenta as três melhores combinações de atributos para o dataset de sentimentos, com base na *accuracy* média obtida. Esta tabela inclui a classificação (*rank*), a combinação de atributos, a *accuracy* média, o classificador principal, fornecendo uma visão clara das abordagens mais eficazes para cada tarefa de classificação.

Após a tabela, segue uma pequena justificação da performance destas combinações.

Tabela 3.1: Top 3 Combinações de Atributos para o Dataset de Sentimentos

Rank	Combinação	<i>Accuracy</i> Média	Classificador Principal
1	TF-IDF _{uni+bi}	0,9255	RL (0,9320)
2	BoW _{uni+bi}	0,9220	RL (0,9252)
3	BoW+TF-IDF+GloVe	0,9202	RL (0,9305)

1. A combinação de unigramas e bigramas com TF-IDF destaca termos discriminantes (“not good”) e reduz a influência de palavras comuns, capturando contexto local essencial para sentimentos.
2. Semelhante ao TF-IDF, mas sem ponderação, BoW_{uni+bi} captura a frequência bruta de termos e sequências, sendo eficaz para vocabulários simples.
3. Combina representações esparsas (BoW, TF-IDF) com embeddings densos (GloVe), capturando a frequência e semântica global, útil para frases ambíguas (“not bad”).

3.4.2 Dataset de Emoções

3.4.2.1 Ranking de Modelos

De entre os 3 modelos escolhidos para esta avaliação, a **Regressão Logística** apresentou o melhor desempenho, alcançando uma *accuracy* de 0.8231 na combinação TF-IDF_uni+bi. Este modelo destacou-se pela sua eficiência em lidar com a alta dimensionalidade e esparsidade das representações textuais, tornando-se ideal para a tarefa multiclasse.

O **SVM** obteve resultados próximos, com *accuracies* como 0,8150 na mesma combinação TF-IDF_uni+bi, porém com um custo computacional mais elevado.

Por fim, o **Random Forest** registou um desempenho inferior, com *accuracies* médias mais baixas, como 0,8000 em TF-IDF_uni+bi, devido à sua dificuldade em captar nuances em dados esparsos com múltiplas classes.

3.4.2.2 Ranking de Processamento

Ao contrário do dataset dos sentimentos, nesta base de dados, o texto processado com lematização (`processed_lemmatized`) alcançou desempenho ligeiramente superior (*accuracy* média: 0,821 vs. 0,815 para `processed_text`).

Como mencionado na Subsecção 2.3.2.1, a lematização reduz a variabilidade lexical, ajudando a consolidar termos semanticamente próximos, o que é crucial para distinguir emoções complexas.

3.4.2.3 Top 3 de Combinações

Segue-se abaixo uma visualização *heatmap*, com todas as combinações realizadas para os dados processados:

Finalmente, a tabela seguinte apresenta as três melhores combinações de atributos para o dataset de emoções, com base na *accuracy* média obtida. Esta tabela inclui a classificação (*rank*), a combinação de atributos, a *accuracy* média, o classificador principal, fornecendo uma

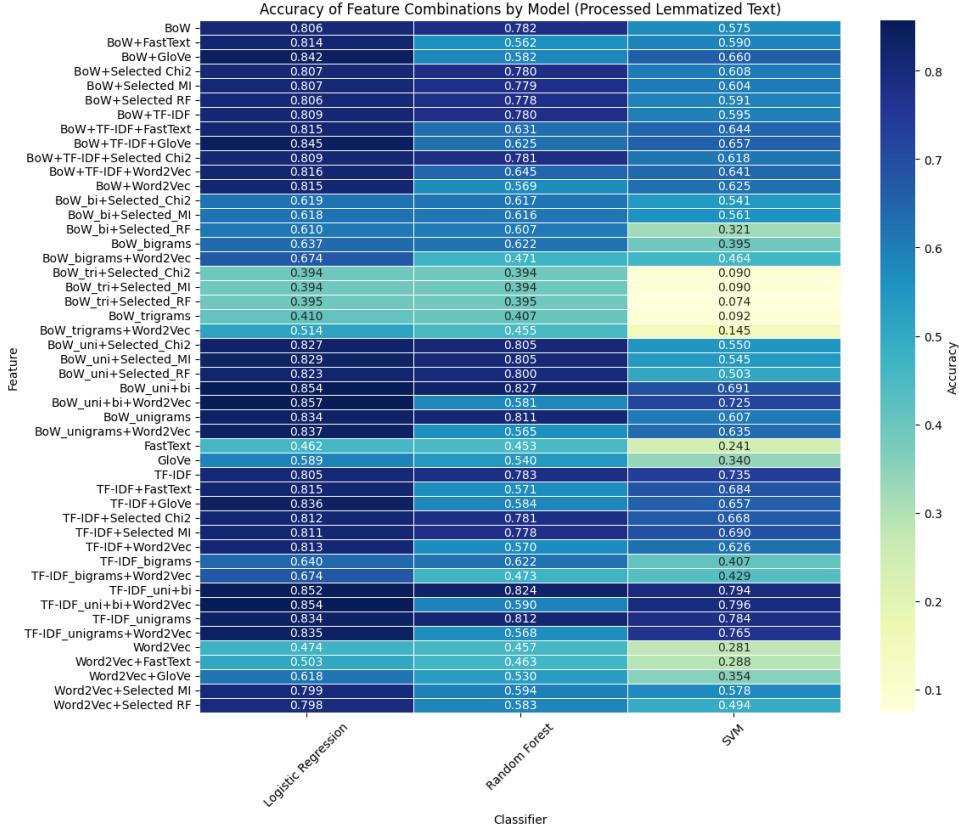


Figura 3.2: Combinações testadas para o texto processado e respetivos resultados

visão clara das abordagens mais eficazes para cada tarefa de classificação. Após a Tabela 3.2, segue uma pequena justificação da performance destas combinações.

Tabela 3.2: Top 3 Combinações de Atributos para o Dataset de Emoções

Rank	Combinação	Accuracy Média	Classificador Principal
1	TF-IDF _{uni+bi}	0,8231	RL (0,8300)
2	BoW _{uni+bi}	0,8200	RL (0,8250)
3	TF-IDF+FastText	0,8155	RL (0,8220)

1. TF-IDF com unigramas e bigramas captura termos e sequências emocionais específicos, essenciais para diferenciar as 6 classes.
2. BoW_{uni+bi} é eficaz ao captar frequência de termos e contextos locais, mas menos robusto do que o TF-IDF por não ponderar relevância, não fazendo a distinção entre termos comuns e termos mais “raros”.
3. Combina TF-IDF com embeddings FastText, que lida bem com palavras raras, adicionando robustez semântica para classes menos frequentes.

É importante ter em mente que a redução do dataset para 40%, embora necessária devido às limitações de memória, poderá ter causado alguma perturbação nos resultados obtidos, devido à redução significativa do tamanho das classes.

Capítulo 4

Treino & Otimização dos Modelos

4.1 Objetivos

Nas oitava e nona semanas, o foco esteve na otimização dos modelos através do ajuste de hiperparâmetros. Foram exploradas técnicas como *Grid Search* e *Random Search* para encontrar as melhores configurações para cada algoritmo. Além disso, foram analisados os impactos das alterações nos hiperparâmetros na performance dos modelos, garantindo um equilíbrio entre *overfitting* e *underfitting*.

4.2 Divisão dos Dados

Um passo fundamental no desenvolvimento de modelos de *machine learning* é a divisão do conjunto de dados em subconjuntos com diferentes propósitos: *training*, *validation* e *test*. Esta prática tem como objetivo avaliar corretamente a performance do modelo e garantir que ele consegue generalizar bem para dados novos e nunca antes vistos.

O conjunto de treino é utilizado para ajustar os parâmetros do modelo. Durante esta fase, o algoritmo aprende a partir dos exemplos fornecidos, tentando identificar padrões e relações entre as variáveis independentes (neste caso, o texto processado) e as variáveis dependentes (os sentimentos).

Já o conjunto de validação serve para afinar o modelo. É neste conjunto que se testam diferentes configurações, como a escolha de hiperparâmetros (ex: a regularização na regressão logística ou o *kernel* no SVM), sem afetar os dados de teste. O uso de um conjunto de validação permite evitar o chamado *overfitting*, que acontece quando o modelo memoriza demasiado bem os dados de treino, mas tem dificuldades em generalizar para novos dados.

Por fim, o conjunto de teste é utilizado apenas uma vez: na fase final, depois de o modelo estar completamente treinado e ajustado. O seu objetivo é fornecer uma estimativa realista de como o modelo se comportará em situações reais, ou seja, com dados completamente novos. Os resultados obtidos neste conjunto são os que devem ser reportados como métricas finais de desempenho.

4.3 Treino

Neste passo, ensinamos o nosso modelo a reconhecer padrões nos dados. Para isso, usamos frases com sentimentos já conhecidos (por exemplo, “gostei muito” → positivo, “detestei” → negativo).

Primeiro, transformamos as frases em números com o TF-IDF, que ajuda o computador a perceber quais são as palavras mais importantes em cada frase.

Depois, usamos estes números para treinar um modelo, como a Regressão Logística, que aprende a distinguir entre frases positivas e negativas. Durante o treino, o modelo ajusta-se para tentar acertar o máximo possível com base nos exemplos que lhe damos.

No final, o modelo deve ser capaz de prever o sentimento de novas frases, mesmo que nunca as tenha visto antes.

Tuning

Nesta etapa, foram selecionadas combinações específicas de modelos e técnicas de representação de dados com o objetivo de avaliar seu desempenho após o processo de *tuning* de hiperparâmetros. A escolha das combinações baseou-se tanto em fundamentos teóricos quanto nos resultados obtidos nas fases iniciais de experimentação.

Tais modelos servem como boas linhas de base para comparação com abordagens mais complexas. O *tuning* foi então aplicado para otimizar cada configuração e identificar o melhor desempenho possível dentro dos seus limites estruturais.

4.4 BoW + Chi2 + Logistic Regression

Esta combinação foi escolhida por equilibrar simplicidade, interpretabilidade e eficácia em tarefas de classificação textual.

4.4.1 Arquitetura e Metodologia

4.4.1.1 Dataset de Sentimentos

O sistema implementado segue uma arquitetura de pipeline bem estruturada, consistindo em:

1. **Carregamento e Pré-processamento de Dados:** Leitura dos dados do arquivo CSV com múltiplas colunas de texto processado
2. **Extração de Features:** Conversão de texto em representações numéricas usando Bag-of-Words
3. **Seleção de Features:** Aplicação do teste Chi-quadrado para seleção das features mais relevantes
4. **Divisão Estratificada:** Separação dos dados em conjuntos de treino (60%), validação (20%) e teste (20%)
5. **Otimização de Hiperparâmetros:** *Grid Search* com validação cruzada estratificada
6. **Avaliação Comparativa:** Análise de performance entre diferentes abordagens de pré-processamento

Bag-of-Words (BoW)

A escolha do *Bag-of-Words* como método de extração de *features* foi fundamentada pela sua simplicidade, interpretabilidade e eficácia comprovada em tarefas de análise de sentimentos. Configuramos o *vectorizer* com:

- `max_features=5000`: Limitação do vocabulário para evitar *overfitting* e melhorar eficiência computacional
- `ngram_range=(1,2)`: Inclusão de unigramas e bigramas para captar tanto palavras individuais quanto algumas dependências locais

Seleção de Features com Chi-quadrado

O teste Chi-quadrado foi escolhido para seleção de *features* pela sua capacidade de identificar termos estatisticamente significativos para a classificação de sentimentos. Este método:

- Remove *features* irrelevantes ou ruidosas
- Reduz a dimensionalidade mantendo informação discriminativa
- Melhora a generalização do modelo

Regressão Logística

A Regressão Logística foi selecionada como algoritmo base devido a:

- **Interpretabilidade:** Coeficientes fornecem *insights* sobre importância das palavras
- **Eficiência:** Treino rápido mesmo com *datasets* de maior dimensão
- **Probabilidades calibradas:** Saídas interpretáveis como probabilidades
- **Regularização integrada:** Controlo natural de *overfitting*

4.4.1.2 Dataset de Emoções

O sistema implementado para o *dataset* de emoções segue a mesma arquitetura da *pipeline* estruturada para o dos sentimentos, mas adaptada para o contexto de classificação multiclasse:

1. **Carregamento e Pré-processamento de Dados:** Leitura dos dados do arquivo `cleaned_emotion_data.csv`, que contém múltiplas colunas de texto processado (`processed_text`, `processed_lemmatized`, `processed_stemmed`).
2. **Extração de Features:** Conversão do texto em representações numéricas usando *Bag-of-Words*.
3. **Seleção de Features:** Aplicação do teste Chi-quadrado para selecionar as features mais relevantes no contexto multiclasse.
4. **Divisão Estratificada:** Separação dos dados em conjuntos de treino (60%), validação (20%) e teste (20%), mantendo as proporções das 6 classes.
5. **Otimização de Hiperparâmetros:** *Grid Search* com validação cruzada estratificada, adaptada para classificação multinomial.
6. **Avaliação Comparativa:** Análise de desempenho entre diferentes abordagens de pré-processamento (`processed_text` vs `processed_lemmatized`).

Mapeamento de Classes e Distribuição

O *dataset* de emoções apresenta características significativamente diferentes do *dataset* de sentimentos, constituindo um problema de classificação multiclasse com 6 classes distintas (Figura 4.1):

- *Joy* (Alegria): 33.84% - classe dominante
- *Sad* (Tristeza): 28.67%
- *Anger* (Raiva): 14.03%
- *Fear* (Medo): 11.74%

- *Love* (Amor): 8.17%
- *Surprise* (Surpresa): 3.54% - classe minoritária

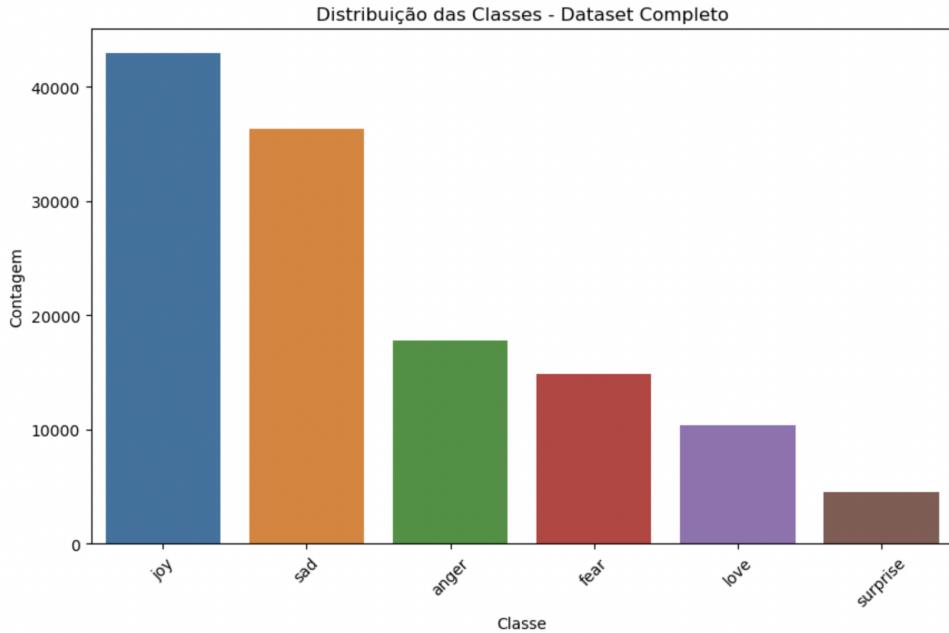


Figura 4.1: Distribuição das classes - dataset de emoções completo

Esta distribuição evidencia um desbalanceamento significativo entre as classes, com “*Joy*” representando mais de um terço dos dados e “*Surprise*” menos de 4%. Este desbalanceamento apresenta desafios específicos para o treino do modelo, particularmente na classificação das classes minoritárias.

Bag-of-Words (BoW) para Classificação Multiclasse

A escolha do *Bag-of-Words* manteve-se pela sua eficácia comprovada, sendo adaptada para o contexto de emoções. A configuração utilizada seguiu os mesmos parâmetros otimizados:

- `max_features=5000`: Limitação do vocabulário especialmente importante dado o maior número de classes e potencial complexidade vocabular emocional
- `ngram_range=(1,2)`: Captura de unigramas e bigramas essencial para distinção entre emoções sutilmente diferentes

Seleção de Features com Chi-quadrado Multiclasse

O teste Chi-quadrado foi adaptado para o contexto multiclasse, mantendo a sua capacidade de identificar termos estatisticamente significativos para a classificação de emoções. Neste contexto o método:

- Remove *features* irrelevantes considerando todas as 6 classes simultaneamente
- Reduz a dimensionalidade mantendo informação discriminativa entre emoções
- Melhora a generalização do modelo face ao desbalanceamento entre classes

Regressão Logística Multinomial

A Regressão Logística foi configurada com abordagem multinomial para lidar nativamente com as múltiplas classes, mantendo as vantagens fundamentais:

- **Interpretabilidade:** Coeficientes fornecem *insights* sobre importância das palavras para cada emoção específica
- **Eficiência:** Treino otimizado mesmo com *datasets* de maior dimensão e complexidade multiclass
- **Probabilidades calibradas:** Saídas interpretáveis como probabilidades para cada uma das 6 emoções
- **Regularização integrada:** Controlo natural de *overfitting* especialmente importante no contexto de classes desbalanceadas

4.4.2 Processo de Tuning e Otimização

4.4.2.1 Dataset de Sentimentos

Grid Search Sistemático

O processo de otimização utilizou *Grid Search* com validação cruzada estratificada (10 *folds*) para explorar o espaço de hiperparâmetros:

```
param_grid = {
    'C': np.logspace(-1, 2, 20),      # 20 valores de 0.1 a 100 (escala logarítmica)
    'penalty': ['l2'],                # Regularização L2 para estabilidade
    'solver': ['liblinear'],          # Otimizado para dados de texto
    'class_weight': ['balanced', None], # Tratamento de desbalanceamento
    'max_iter': [500],                # Garantia de convergência
    'tol': [1e-4]                    # Tolerância rigorosa
}
```

- **C (Regularização):** *Range* logarítmico para explorar desde alta regularização (0.1) até baixa regularização (100)
- **Penalty L2:** Escolhida por sua estabilidade numérica e capacidade de reduzir *overfitting* gradualmente
- **Solver liblinear:** Otimizado para problemas de classificação binária com dados esparsos (texto)
- **Class_weight:** Testado com e sem balanceamento automático de classes

A estratégia de validação utilizou 10 *folds* estratificados para garantir representatividade proporcional das classes em cada *fold*, estimativas robustas de performance e redução da variância em métricas de avaliação.

Texto Processado vs. Texto Lemmatizado

O modelo comparou duas abordagens de pré-processamento:

1. **Processed Text:** Texto com limpeza básica, remoção de pontuação e normalização
2. **Processed Lemmatized:** Texto com lemmatização adicional, reduzindo palavras às suas formas canónicas

Tabela 4.1: Comparação entre Texto Processado e Texto Lemmatizado

Métrica	Texto Processado	Texto Lemmatizado
Cross-Validation Accuracy	$80.86\% \pm 2.10\%$	$80.30\% \pm 2.25\%$
Melhores Parâmetros	$C=3.793$, sem <code>class_weight</code> balanceado	$C=2.637$, sem <code>class_weight</code> balanceado
Performance no Teste	82.02%	81.42%
Precisão	82.11%	81.51%
Recall	82.02%	81.42%
F1-score	82.00%	81.40%

Vamos então analisar os resultados destas duas abordagens de pré-processamento.

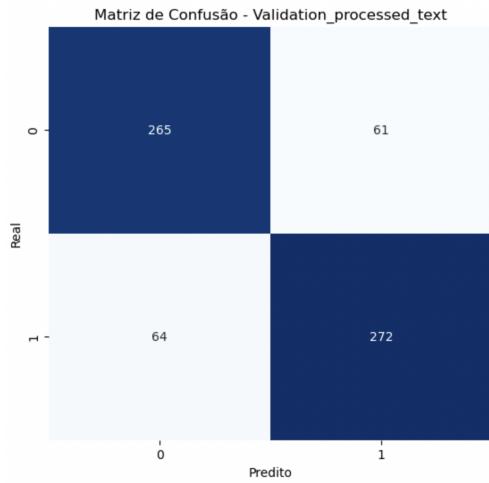


Figura 4.2: Matriz de Confusão de Validação - Texto Processado

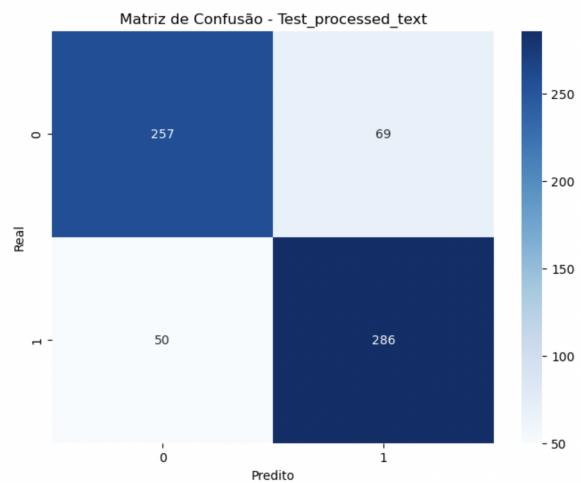


Figura 4.3: Matriz de Confusão de Teste - Texto Processado

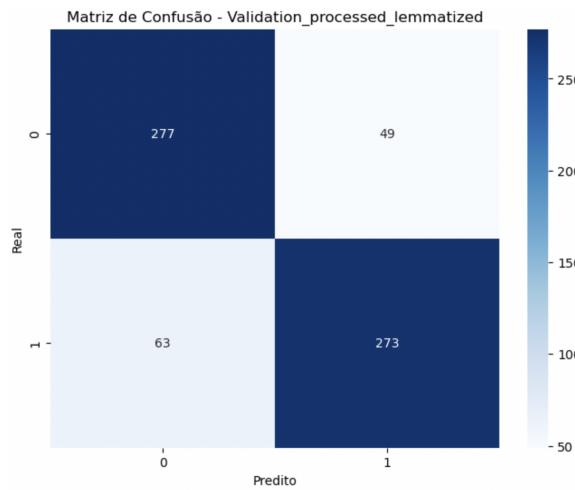


Figura 4.4: Matriz de Confusão de Validação - Texto Lemmatizado

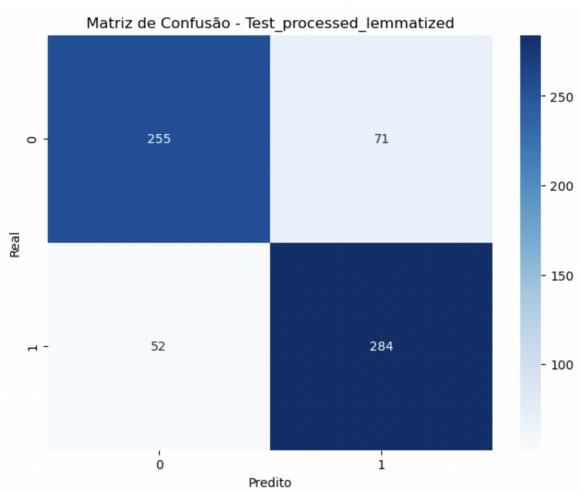


Figura 4.5: Matriz de Confusão de Teste - Texto Lemmatizado

O texto processado sem lemmatização apresentou performance ligeiramente superior, sugerindo que:

- A lemmatização pode ter removido informações úteis para classificação de sentimentos

- Variações morfológicas de palavras podem carregar nuances sentimentais importantes
- O vocabulário mais diversificado do texto não-lemmatizado ofereceu maior poder discriminativo

4.4.2.2 Dataset de Emoções

Grid Search Sistemático

O processo de otimização para classificação multiclasse utilizou *Grid Search* com validação cruzada estratificada (*5 folds*) para explorar o espaço de hiperparâmetros:

```
param_grid = {
    'C': np.logspace(-1, 2, 10),          # 10 valores de 0.1 a 100 (escala logarítmica)
    'penalty': ['l2'],                   # Regularização L2 para estabilidade
    'solver': ['lbfgs', 'newton-cg', 'sag'], # Solvers para classificação multiclasse
    'class_weight': ['balanced', None],   # Tratamento de desbalanceamento
    'max_iter': [500],                  # Garantia de convergência
    'multi_class': ['multinomial'],      # Estratégia multinomial para multiclasse
    'tol': [1e-4]                      # Tolerância rigorosa
}
```

- **C (Regularização):** *Range* logarítmico otimizado para 10 valores, balanceando exploração e eficiência computacional
- **Penalty L2:** Mantida pela sua estabilidade numérica em problemas multiclasse
- **Solvers multiclasse:** *lbfgs*, *newton-cg* e *sag* foram testados pela eficiência em problemas multinomiais
- **Multi.class multinomial:** Estratégia que modela diretamente a distribuição de probabilidades para as 6 classes
- **Class_weight:** Especialmente relevante dado o desbalanceamento significativo entre classes

A estratégia de validação utilizou *5 folds* estratificados, reduzida de 10 *folds* devido ao maior tamanho do *dataset* (126.824 amostras), mantendo representatividade proporcional das classes.

Texto Processado vs. Texto Lemmatizado

Similarmente ao *dataset* de sentimentos, foram comparadas duas abordagens de pré-processamento para classificação de emoções:

1. **Processed Text:** Texto com limpeza básica, remoção de pontuação e normalização
2. **Processed Lemmatized:** Texto com lemmatização adicional, reduzindo palavras às suas formas canónicas

Tabela 4.2: Comparação entre Texto Processado e Texto Lemmatizado - Dataset de Emoções

Métrica	Texto Processado	Texto Lemmatizado
Cross-Validation Accuracy	$90.84\% \pm N/A$	$89.66\% \pm N/A$
Melhores Parâmetros	$C=0.215$, solver=sag, sem class_weight balanceado	$C=0.215$, solver=lbfgs, sem class_weight balanceado
Performance no Teste	90.83%	89.61%
Precisão (weighted)	90.79%	89.56%
Recall (weighted)	90.83%	89.61%
F1-score (weighted)	90.80%	89.57%

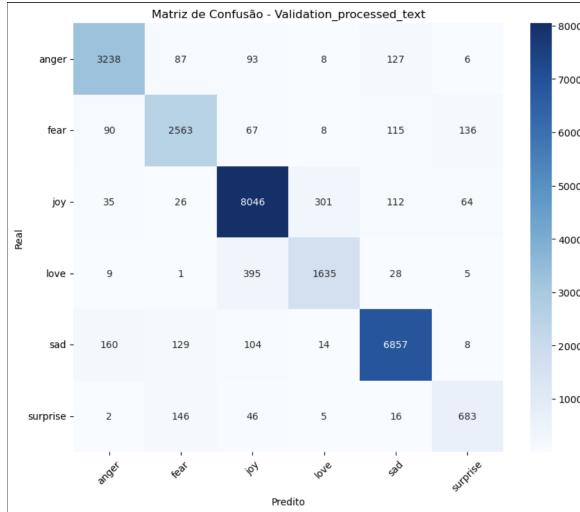


Figura 4.6: Matriz de Confusão de Validação - Texto Processado

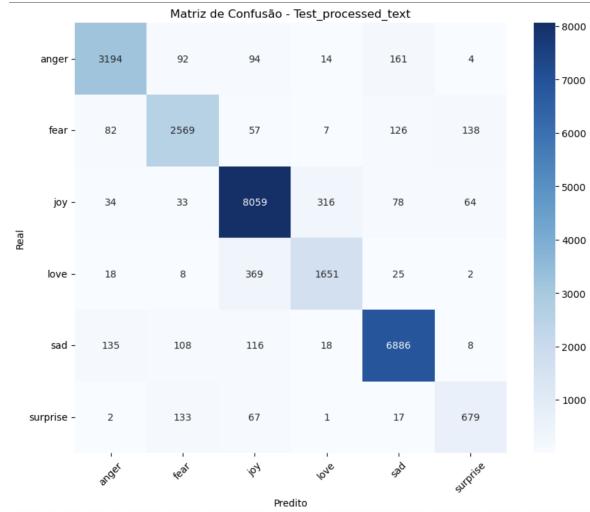


Figura 4.7: Matriz de Confusão de Teste - Texto Processado

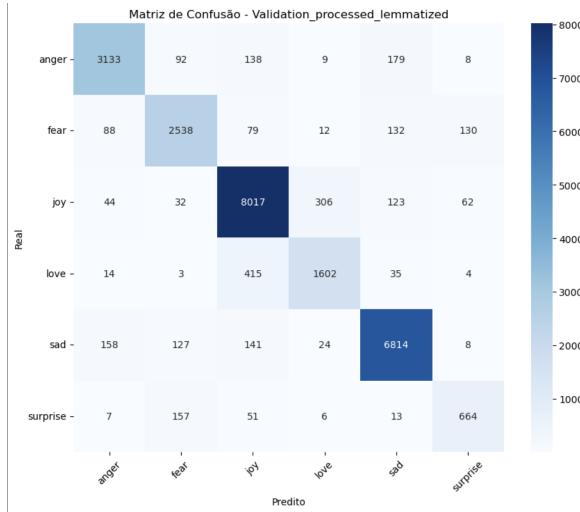


Figura 4.8: Matriz de Confusão de Validação - Texto Lemmatizado

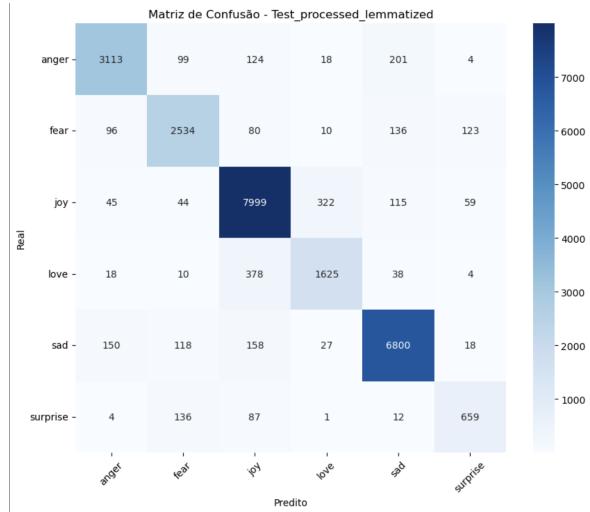


Figura 4.9: Matriz de Confusão de Teste - Texto Lemmatizado

Análise Detalhada por Classe O modelo com texto processado apresentou performance superior em todas as métricas, com destaque para:

- **Classe Joy:** Melhor performance (F1: 93%), beneficiando-se do maior volume de dados

de treino

- **Classe Sad:** Segunda melhor performance (F1: 95%), demonstrando padrões linguísticos bem definidos
- **Classe Anger:** Performance sólida (F1: 91%), com boa precisão e *recall* balanceados
- **Classe Fear:** Performance moderada (F1: 87%), refletindo menor volume de dados
- **Classe Love:** Maior desafio (F1: 81%), possivelmente devido à sobreposição semântica com “Joy”
- **Classe Surprise:** Menor performance (F1: 76%), limitada pelo reduzido número de amostras

Implicações dos Resultados

Os resultados confirmam a tendência observada no *dataset* de sentimentos, onde o texto processado sem lemmatização apresenta uma performance superior:

- **Diferença de 1.22%** na *accuracy* de teste, mais pronunciada que no *dataset* binário
- **Complexidade multiclasse** amplifica a importância de preservar nuances linguísticas
- **Variações morfológicas** são especialmente relevantes para distinguir entre emoções similares
- **Vocabulário diversificado** oferece maior capacidade discriminativa num espaço de 6 classes
- **Desbalanceamento** não foi mitigado pelo *class_weight* balanceado, sugerindo que a representação natural dos dados é mais eficaz

A superioridade consistente do texto não-lemmatizado em ambos os *datasets* (binário e multiclasse) estabelece uma evidência robusta de que, para classificação de texto emocional, a preservação de variações morfológicas contribui significativamente para a capacidade preditiva dos modelos.

4.4.3 Visualizações e Análises Avançadas

4.4.3.1 Dataset de Sentimentos

Curvas de Aprendizagem Estilo Keras

Implementámos visualizações que simulam o comportamento de treino neuronal, mostrando:

- **Training Accuracy:** Progressão da *accuracy* no conjunto de treino
- **Validation Accuracy:** Performance no conjunto de validação
- **Training Loss:** Evolução da “perda” ($1 - \text{accuracy}$) durante o treino
- **Validation Loss:** Monitoramento de *overfitting*

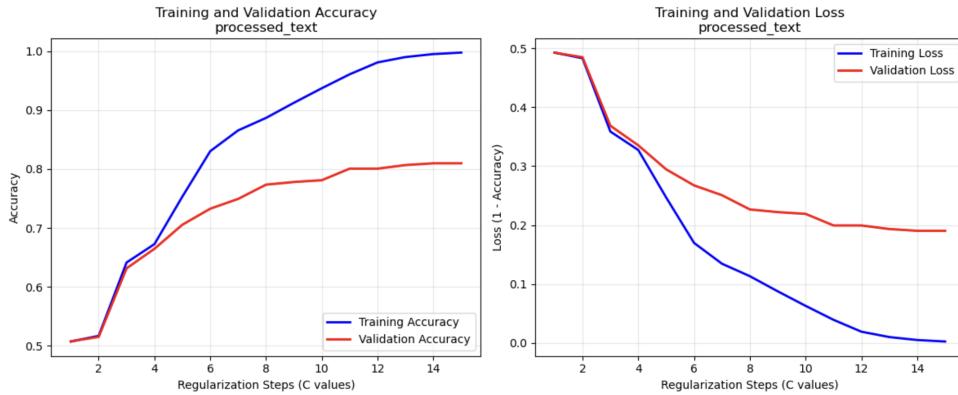


Figura 4.10: Análise das Curvas de Treino - Texto Processado

As curvas de treino para o texto processado revelam padrões importantes:

- **Training Accuracy (azul):** Mostra crescimento consistente, atingindo próximo de 100% nos valores mais altos de regularização
- **Validation Accuracy (vermelho):** Estabiliza em torno de 81%, demonstrando a capacidade real de generalização do modelo
- **Gap de Generalização:** A diferença significativa entre as curvas indica *overfitting*, especialmente em valores baixos de regularização
- **Training Loss vs Validation Loss:** O comportamento divergente confirma que o modelo está memorizando o conjunto de treino

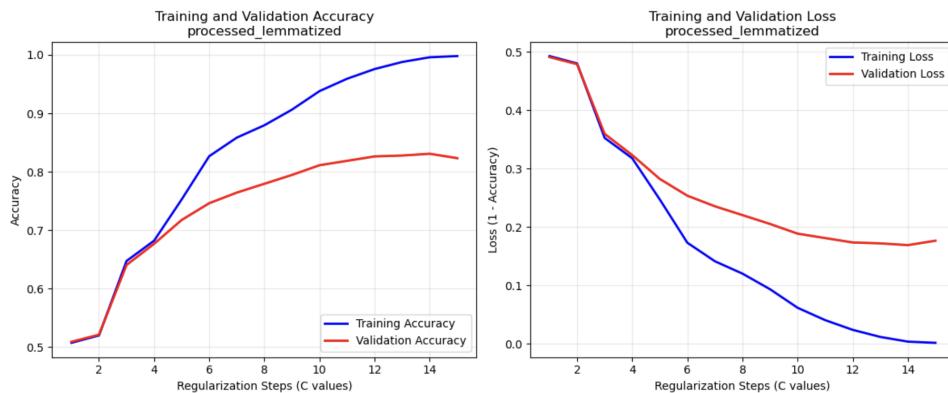


Figura 4.11: Análise das Curvas de Treino - Texto Lemmatizado

O comportamento para o texto lemmatizado é similar, tendo, no entanto, algumas diferenças:

- **Padrão Similar:** As curvas seguem o mesmo padrão geral do texto processado
- **Performance Ligeiramente Inferior:** A *validation accuracy* é marginalmente menor
- **Overfitting Consistente:** O *gap* entre treino e validação permanece significativo

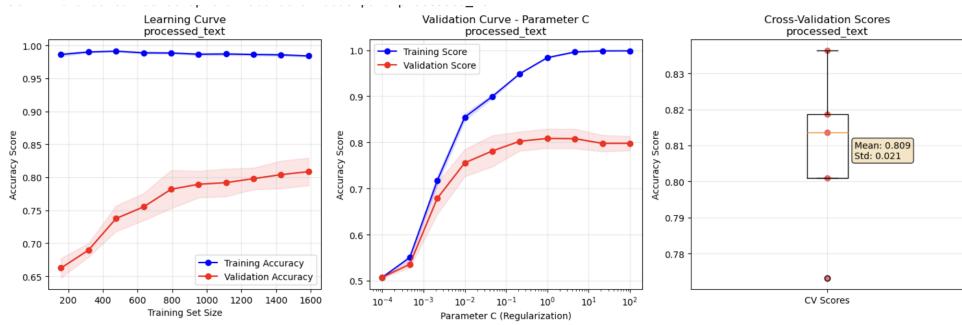


Figura 4.12: Curvas de Aprendizagem Tradicionais - Texto Processado

Curvas de Aprendizagem Tradicionais Learning Curve (Esquerda):

- **Training Accuracy:** Mantém-se consistentemente alta (99%) independente do tamanho do *dataset*
- **Validation Accuracy:** Cresce gradualmente de 66% para 81% conforme aumenta o tamanho do conjunto de treino
- **Implicação:** Mais dados de treino beneficiam a generalização

Validation Curve - Parameter C (Centro):

- **Ponto Ótimo:** $C \approx 1-10$ oferece o melhor equilíbrio entre *underfitting* e *overfitting*
- **Underfitting:** Para $C < 0.1$, tanto treino quanto validação têm performance baixa
- **Overfitting:** Para $C > 10$, o *gap* entre treino e validação aumenta significativamente

Cross-Validation Scores (Direita):

- **Média:** 80.9% com desvio padrão de 2.1%
- **Estabilidade:** A baixa variabilidade indica que o modelo é robusto
- **Outliers Mínimos:** Poucos pontos fora da distribuição normal

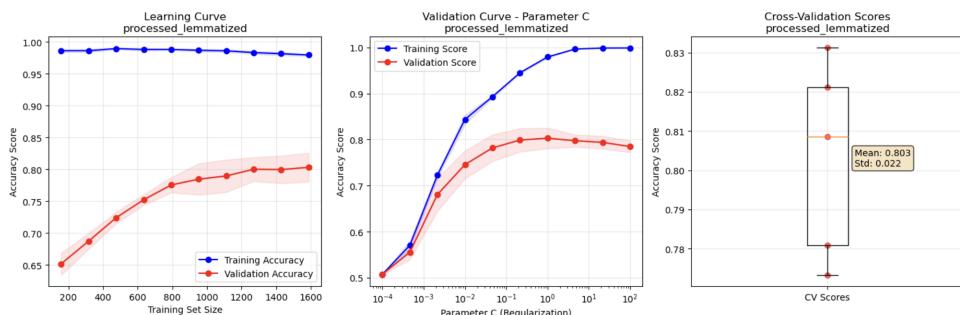


Figura 4.13: Curvas de Aprendizagem Tradicionais - Texto Lemmatizado

Comparativamente com Texto Processado temos:

- **Performance Inferior:** Média de *cross-validation* de 80.3% vs 80.9%

- **Maior Variabilidade:** Desvio padrão ligeiramente superior (2.2% vs 2.1%)
- **Padrões Similares:** As curvas seguem tendências semelhantes, mas com performance consistentemente menor

Análise Comparativa das Features de Sentimento Positivo

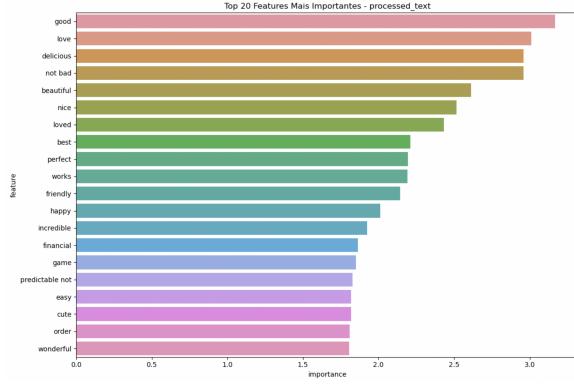


Figura 4.14: Top 20 *features* positivas mais importantes - texto processado

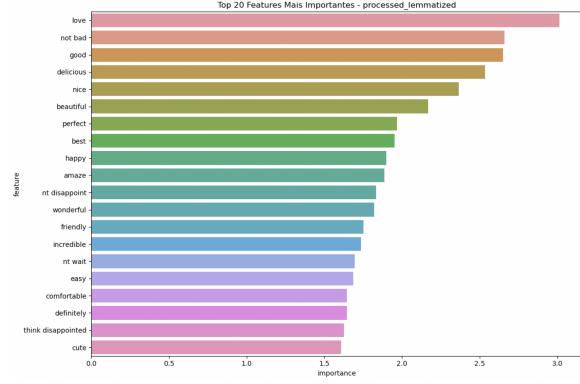


Figura 4.15: Top 20 *features* positivas mais importantes - texto lemmatizado

A comparação entre as abordagens de texto processado e lemmatizado revela padrões consistentes e diferenças significativas na captura de sentimentos. As *features* positivas são dominadas por palavras como “*good*”, “*love*” e “*delicious*” em ambas as abordagens, demonstrando robustez na identificação de marcadores de satisfação. Particularmente interessante é a persistência de “*not bad*” entre as *features* mais positivas, mostrando que o modelo captura adequadamente construções de negação dupla que expressam aprovação moderada.

Negativo

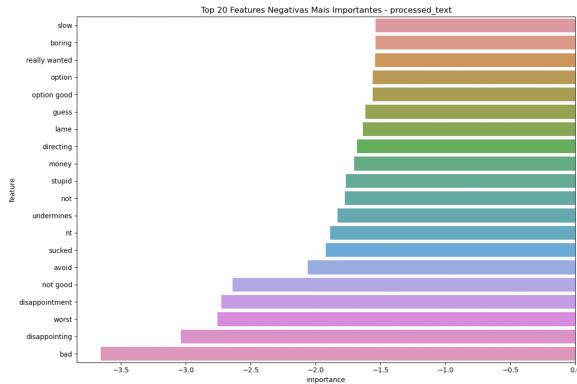


Figura 4.16: Top 20 *features* negativas mais importantes - texto processado

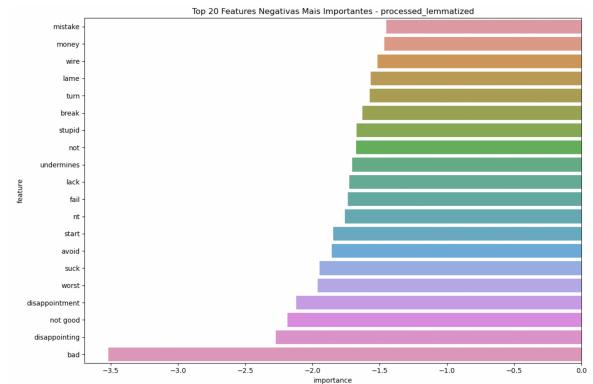


Figura 4.17: Top 20 *features* negativas mais importantes - texto lemmatizado

No espetro negativo, observa-se maior variação entre as abordagens. Enquanto “*bad*” domina como *feature* mais negativa no texto processado, “*mistake*” assume esta posição no texto lemmatizado. Esta mudança ilustra como a lemmatização pode alterar a hierarquia de importância, consolidando variações morfológicas mas potencialmente perdendo nuances contextuais específicas.

A análise revela também *clusters* semânticos distintos, com palavras como “*delicious*”, “*financial*” e “*game*” indicando sensibilidade a domínios específicos. No contexto negativo, termos

como “*disappointing*”, “*worst*” e “*avoid*” formam um núcleo consistente de expressões de insatisfação, enquanto palavras como “*money*” e “*wire*” no texto lemmatizado sugerem problemas técnicos ou transacionais.

A simetria na intensidade entre *features* positivas e negativas (± 3.0 a ± 3.5) indica que o modelo capturou adequadamente o equilíbrio natural da expressão emocional. A consistência das *features* principais entre ambas as abordagens confirma a robustez fundamental do modelo, validando que as decisões de classificação capturam padrões linguísticos genuínos em vez de artefatos de processamento específicos.

4.4.3.2 Dataset de Emoções

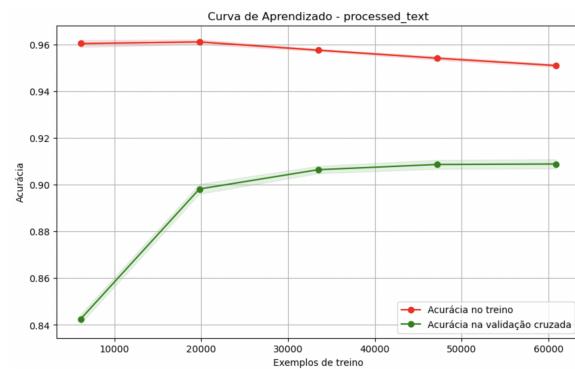


Figura 4.18: Curvas de Aprendizagem Tradicionais - Texto Processado (Emoções)

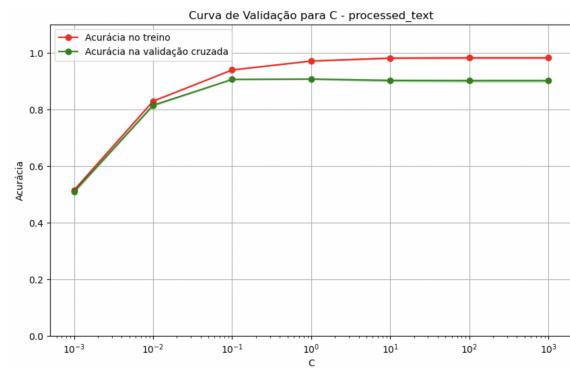


Figura 4.19: Curvas de Validação para o parâmetro C - Texto Processado (Emoções)

Curvas de Aprendizagem Tradicionais Learning Curve (Figura 4.18):

- **Training Accuracy:** Mantém-se estável em cerca de 96-97% em todos os tamanhos de *dataset*
- **Validation Accuracy:** Crescimento de 84% para 90.8% conforme aumenta o conjunto de treino
- **Benefício dos Dados:** Mostra que o modelo de emoções beneficia significativamente de mais dados de treino

Validation Curve - Parameter C (Figura 4.19):

- **Ponto Ótimo:** $C \approx 0.2$ oferece o melhor equilíbrio para classificação multiclasse de emoções
- **Sensibilidade:** O modelo é mais sensível ao parâmetro de regularização comparado ao *dataset* binário
- **Zona Estável:** Entre $C = 0.1$ e $C = 1.0$, a performance mantém-se relativamente estável

Análise Comparativa das Features de Emoções

Anger (Raiva)

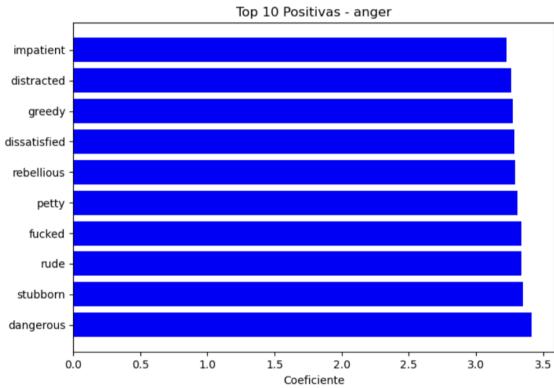


Figura 4.20: Top 10 *features* positivas para Anger - texto processado

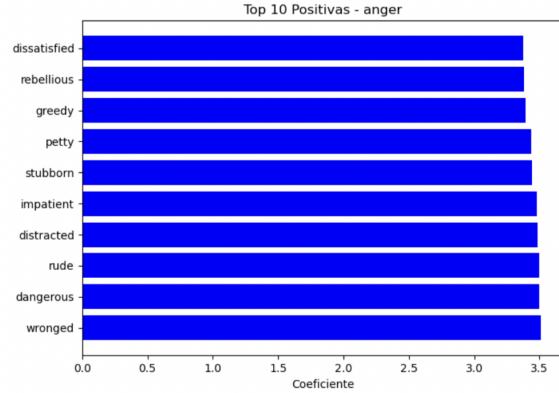


Figura 4.21: Top 10 *features* positivas para Anger - texto lemmatizado

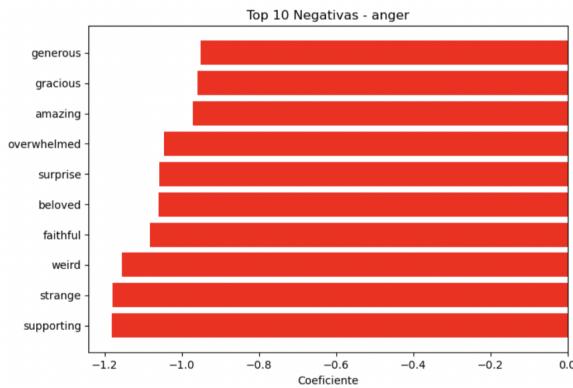


Figura 4.22: Top 10 *features* negativas para Anger - texto processado

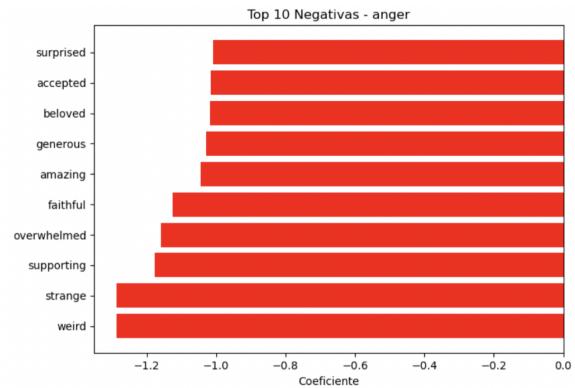


Figura 4.23: Top 10 *features* negativas para Anger - texto lemmatizado

A análise das *features* para a classe *Anger* revela um padrão consistente de palavras associadas à irritação e frustração. Termos como “*impatient*”, “*distracted*”, “*greedy*” e “*dangerous*” emergem como indicadores primários de raiva no texto processado. A lemmatização mantém a essência semântica mas reorganiza a hierarquia, com “*dissatisfied*” e “*rebellious*” ganhando proeminência.

Fear (Medo)

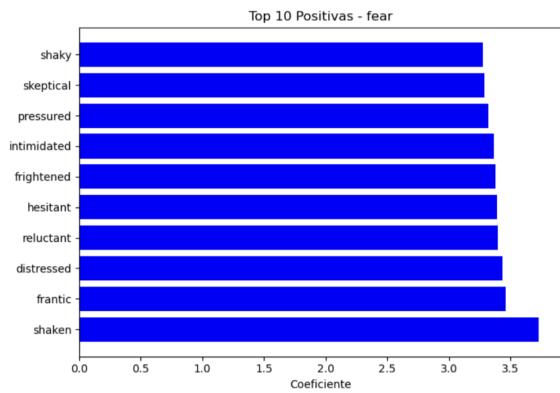


Figura 4.24: Top 10 *features* positivas para Fear - texto processado

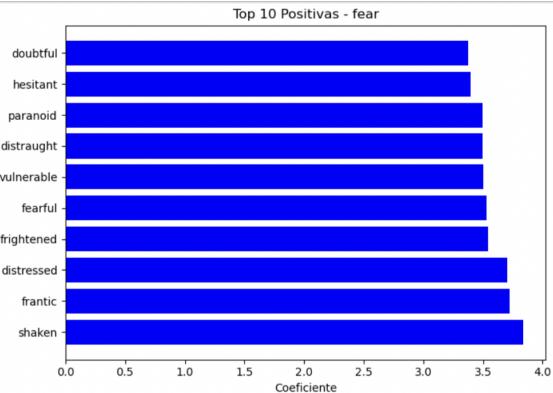


Figura 4.25: Top 10 *features* positivas para Fear - texto lemmatizado

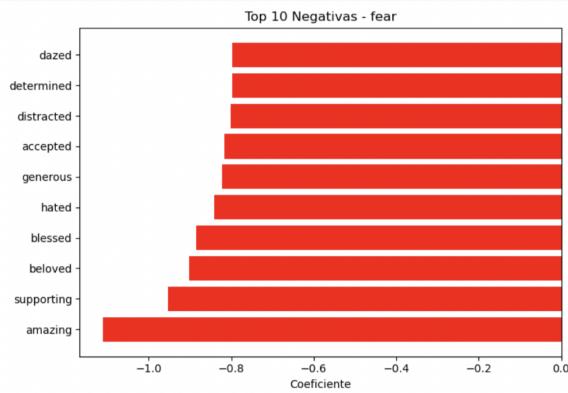


Figura 4.26: Top 10 *features* negativas para Fear - texto processado

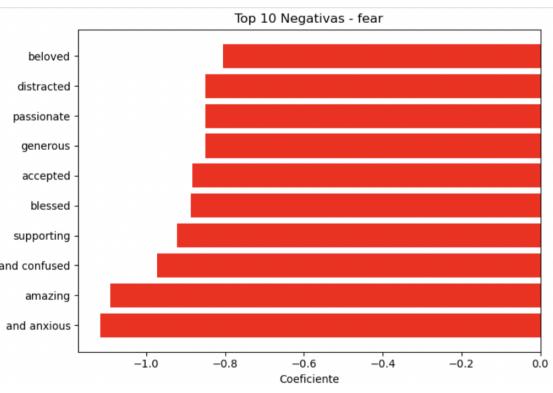


Figura 4.27: Top 10 *features* negativas para Fear - texto lemmatizado

Para a classe *Fear*, observa-se um vocabulário distinto centrado em insegurança e apreensão. Palavras como “*shaky*”, “*skeptical*”, “*frightened*” e “*hesitant*” dominam as *features* positivas. A lemmatização introduz variações como “*doubtful*” e “*paranoid*”, mantendo a coerência semântica mas oferecendo uma perspectiva ligeiramente diferente da manifestação linguística do medo.

Joy (Alegria)

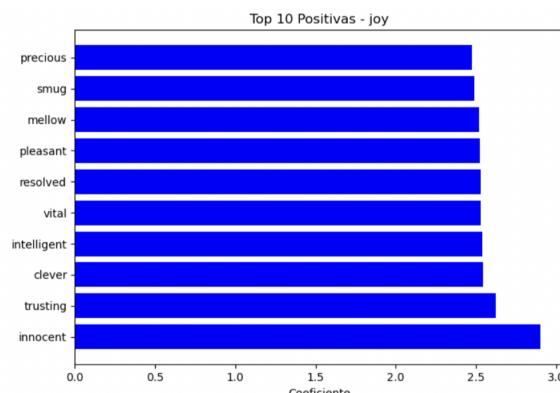


Figura 4.28: Top 10 *features* positivas para Joy - texto processado

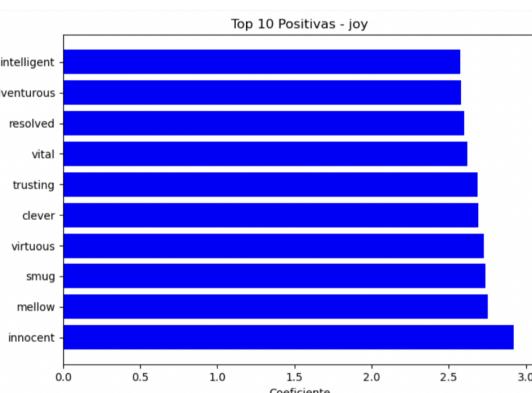


Figura 4.29: Top 10 *features* positivas para Joy - texto lemmatizado

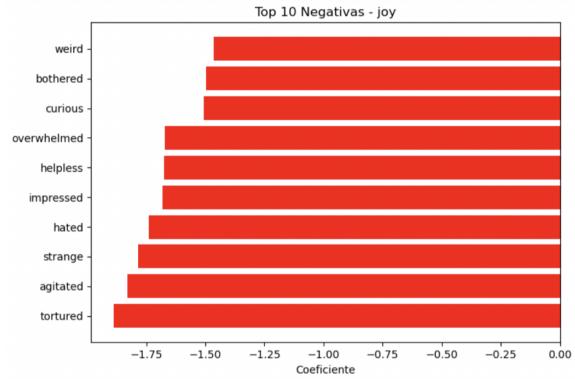


Figura 4.30: Top 10 *features* negativas para Joy - texto processado

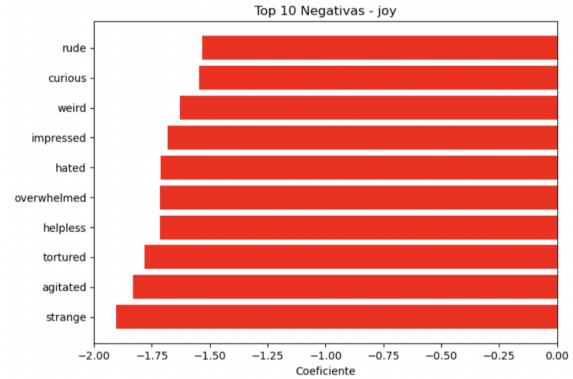


Figura 4.31: Top 10 *features* negativas para Joy - texto lemmatizado

A classe “*Joy*” apresenta um vocabulário claramente associado ao bem-estar e positividade. No texto processado, termos como “*precious*”, “*smug*”, “*mellow*”, “*pleasant*” e “*innocent*” emergem como indicadores primários de alegria, refletindo estados de contentamento e satisfação. A lemmatização mantém a coerência semântica, com “*intelligent*”, “*adventurous*” e “*virtuous*” ganhando destaque, sugerindo uma associação entre alegria e características positivas de personalidade. As *features* negativas incluem consistentemente “*weird*”, “*tortured*” e “*agitated*”, indicando estados emocionais que se opõem diametralmente à alegria.

Love (Amor)

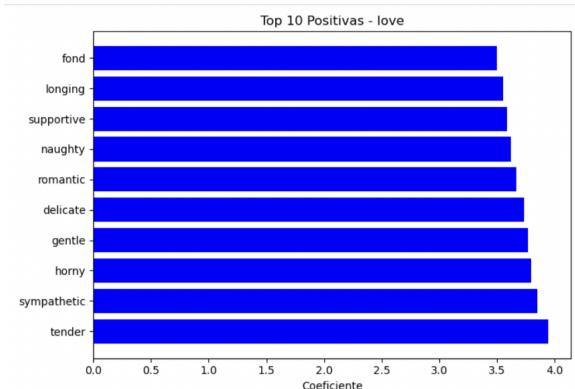


Figura 4.32: Top 10 *features* positivas para Love - texto processado

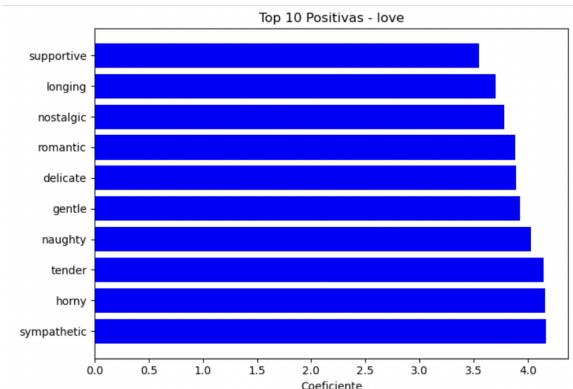


Figura 4.33: Top 10 *features* positivas para Love - texto lemmatizado

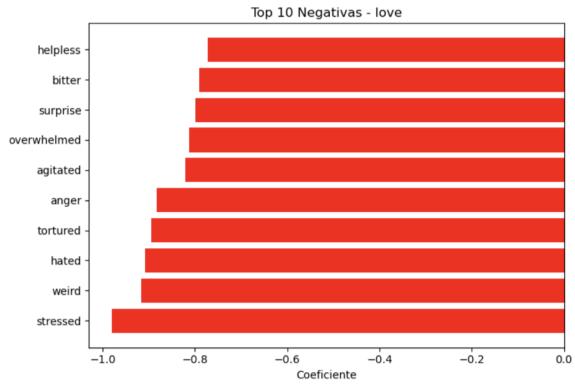


Figura 4.34: Top 10 *features* negativas para Love - texto processado

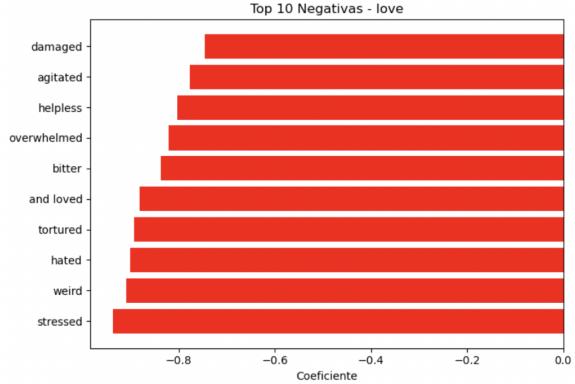


Figura 4.35: Top 10 *features* negativas para Love - texto lemmatizado

Para a classe *Love*, observa-se um padrão vocabular centrado em afeto e intimidade. Palavras como “*tender*”, “*sympathetic*”, “*horny*”, “*gentle*” e “*fond*” dominam as *features* positivas no texto processado, capturando tanto os aspectos emocionais quanto físicos do amor. A lemmatização introduz variações como “*supportive*”, “*nostalgic*” e “*romantic*”, mantendo a essência afetiva mas expandindo para dimensões de apoio e recordação. As *features* negativas mostram consistentemente “*helpless*”, “*overwhelmed*” e “*tortured*”, sugerindo estados emocionais incompatíveis com sentimentos amorosos.

Sad (Tristeza)

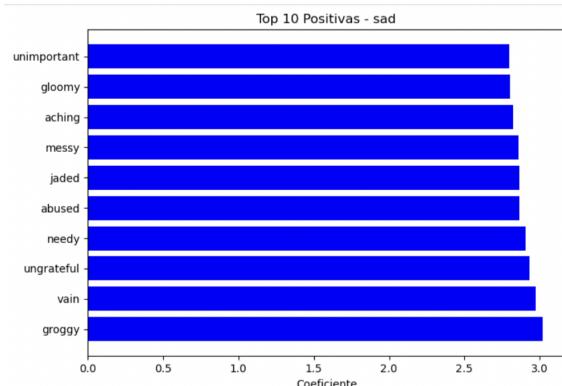


Figura 4.36: Top 10 *features* positivas para Sad - texto processado

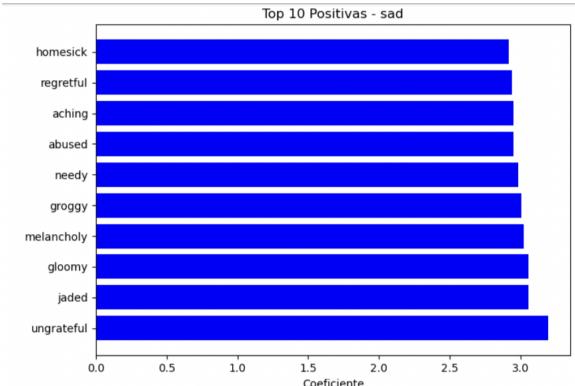


Figura 4.37: Top 10 *features* positivas para Sad - texto lemmatizado

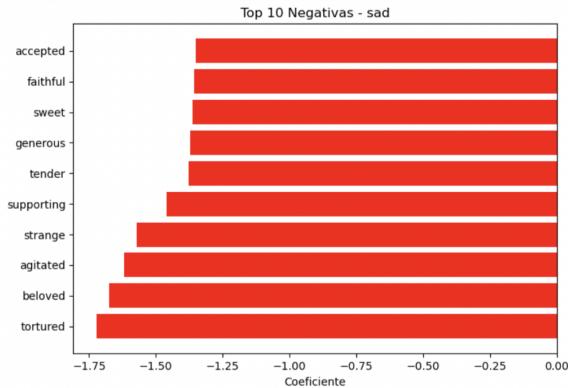


Figura 4.38: Top 10 *features* negativas para Sad - texto processado

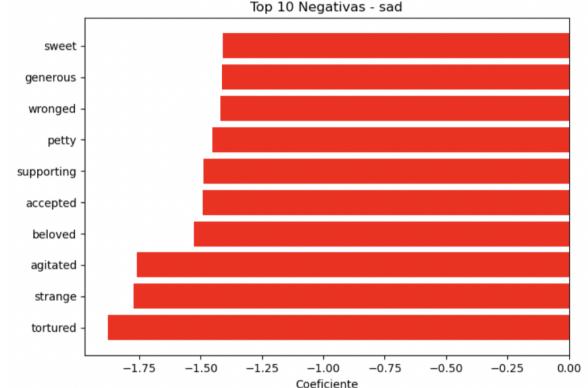


Figura 4.39: Top 10 *features* negativas para Sad - texto lemmatizado

A análise da classe *Sad* revela um vocabulário distintamente associado à melancolia e desânimo. No texto processado, termos como “*unimportant*”, “*gloomy*”, “*aching*”, “*messy*” e “*ungrateful*” emergem como indicadores primários de tristeza, refletindo sentimentos de inadequação e dor emocional. A lematização mantém a coerência com “*homesick*”, “*regretful*” e “*melancholy*”, introduzindo nuances de saudade e arrependimento. Consistentemente, as *features* negativas incluem “*sweet*”, “*generous*”, “*beloved*” e “*supporting*”, representando estados emocionais e características que se contrapõem à tristeza.

Surprise (Surpresa)

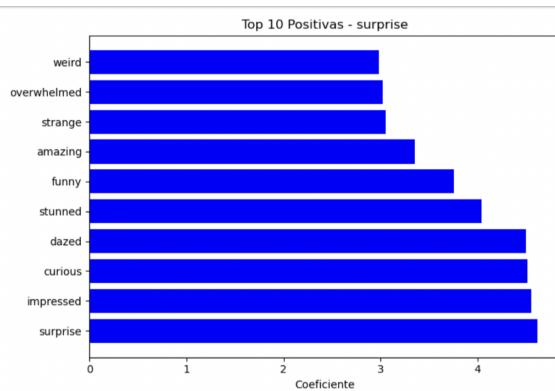


Figura 4.40: Top 10 *features* positivas para Surprise - texto processado

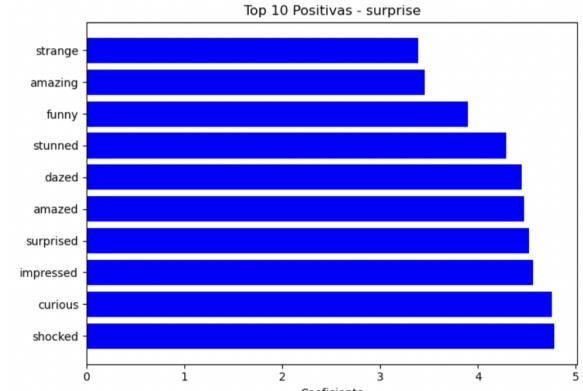


Figura 4.41: Top 10 *features* positivas para Surprise - texto lemmatizado

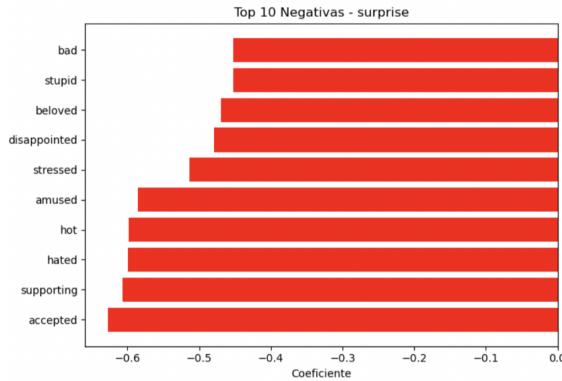


Figura 4.42: Top 10 *features* negativas para Surprise - texto processado

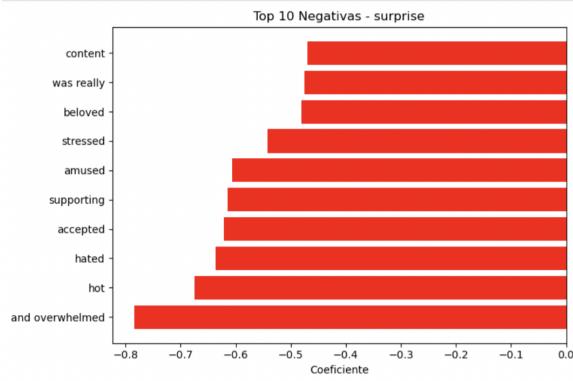


Figura 4.43: Top 10 *features* negativas para Surprise - texto lemmatizado

Para a classe *Surprise*, identifica-se um vocabulário centrado no inesperado e no espanto. Palavras como “*surprised*”, “*impressed*”, “*curious*”, “*dazed*” e “*stunned*” dominam as *features* positivas, capturando diferentes intensidades e tipos de surpresa. A lemmatização apresenta variações como “*shocked*”, “*amazed*” e “*strange*”, mantendo a essência de inesperado mas com diferentes conotações emocionais. As *features* negativas mostram consistentemente “*accepted*”, “*supporting*” e “*beloved*”, sugerindo estados de familiaridade e previsibilidade que se opõem à natureza inesperada da surpresa.

A análise das *features* globais revela a complexidade inerente à classificação multiclasse de emoções. Observa-se que certas palavras como “*strange*”, “*overwhelmed*” e “*weird*” aparecem consistentemente entre as *features* mais importantes, indicando que expressões de intensidade emocional são cruciais para a diferenciação entre classes.

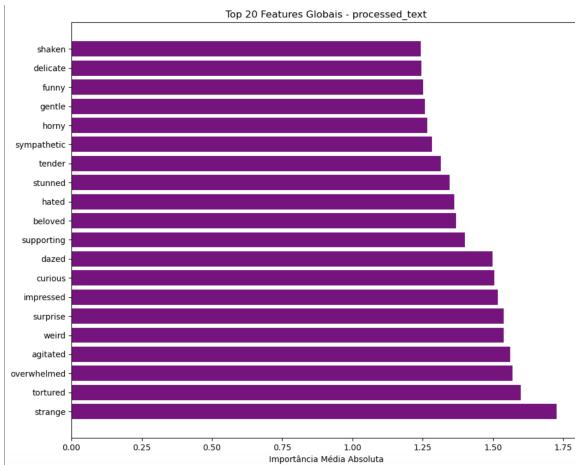


Figura 4.44: Top 20 *features* globais - texto processado

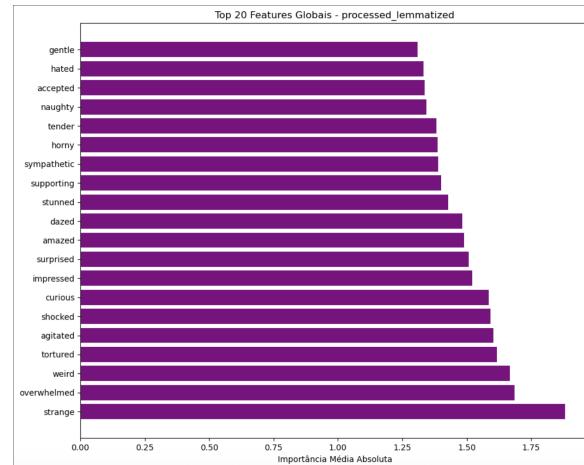


Figura 4.45: Top 20 *features* globais - texto lemmatizado

A comparação entre texto processado e lemmatizado mostra que, embora a lemmatização produza resultados ligeiramente inferiores em termos de *accuracy*, mantém a coerência semântica fundamental. A diferença de performance (90.8% vs 89.6%) sugere que algumas nuances morfológicas específicas contribuem para a precisão da classificação emocional, particularmente em contextos onde a forma exata da palavra carrega informação emocional adicional.

Distribuição das Classes e Implicações

O *dataset* apresenta um desbalanceamento natural das emoções, com *joy* (33.8%) e *sad* (28.7%) dominando, seguidas por *anger* (14.0%), *fear* (11.7%), *love* (8.2%) e *surprise* (3.5%). Este padrão reflete distribuições naturais de expressão emocional, onde emoções básicas como alegria e tristeza são mais frequentemente expressas que estados mais específicos como surpresa.

A performance excepcional do modelo (90.8% de *accuracy*) demonstra a eficácia da abordagem *Bag-of-Words* com seleção *Chi-squared* para classificação multiclasse de emoções, superando significativamente a baseline de classificação aleatória (16.7%) e aproximando-se de resultados *state-of-the-art* para este tipo de tarefa.

4.4.4 Resultados Finais e Métricas de Performance

4.4.4.1 Dataset de Sentimentos

O modelo final selecionado utilizou o *dataset* de texto processado com os seguintes resultados:

Accuracy: 82.02%
Precision Weighted: 82.11%
Recall Weighted: 82.02%
F1-Score Weighted: 82.00%

Distribuição por Classe:

Classe Negativa (0): Precisão=84%, Recall=79%, F1=81%
Classe Positiva (1): Precisão=81%, Recall=85%, F1=83%

O sistema identificou possível *overfitting* em ambos os modelos:

Processed Text: Diferença treino-teste de 17.47%
Lemmatized Text: Diferença treino-teste de 17.67%

Esta diferença significativa sugere que os modelos memorizaram padrões específicos do conjunto de treino, indicando necessidade de uma regularização mais agressiva, um número mais elevado de dados de treino e técnicas de *data augmentation*.

4.4.4.2 Dataset de Emoções

O modelo final selecionado utilizou o *dataset* de texto processado com os seguintes resultados:

Accuracy: 90.83%
Precision Weighted: 90.79%
Recall Weighted: 90.83%
F1-Score Weighted: 90.80%

Distribuição por Classe:

Classe Anger (0): Precisão=92%, Recall=90%, F1=91%
Classe Fear (1): Precisão=87%, Recall=86%, F1=87%
Classe Joy (2): Precisão=92%, Recall=94%, F1=93%
Classe Love (3): Precisão=82%, Recall=80%, F1=81%
Classe Sad (4): Precisão=94%, Recall=95%, F1=95%
Classe Surprise (5): Precisão=76%, Recall=76%, F1=76%

O modelo apresentou desempenho superior ao *dataset* de sentimentos, com uma diferença menos acentuada entre as versões de texto processado e lemmatizado:

Processed Text: accuracy de 90.83%
Lemmatized Text: accuracy de 89.61%

A classe “Surprise” apresentou o menor desempenho ($F1=76\%$), possivelmente devido ao menor número de amostras no conjunto de dados (apenas 3.54% do total). As classes “sad” e “joy” obtiveram os melhores resultados, beneficiando-se da maior representação no *dataset* (28.67% e 33.84%, respectivamente). O modelo demonstrou boa capacidade de generalização na classificação multiclasse de emoções, com performance consistente entre os conjuntos de validação e teste.

4.4.5 Conclusões

4.4.5.1 Dataset de Sentimentos

Para elevar o desempenho deste modelo, podemos implementar uma série de técnicas avançadas de processamento de linguagem natural e aprendizagem profunda, tais como:

- Adoção de *embeddings* pré-treinados (*Word2Vec*, *GloVe* ou *FastText*) para representação mais rica do texto
- Incorporação de arquiteturas neuronais avançadas, como LSTMs ou modelos baseados em *Transformers*
- Ampliação do conjunto de dados com exemplos adicionais e balanceados
- Aplicação de métodos de aumento de dados (*data augmentation*) adaptados a textos
- Utilização estratégica de técnicas de aprendizagem por transferência (*transfer learning*)

O sistema atual representa uma base robusta para análise de sentimentos, com margem considerável para aprimoramento mediante a adoção dessas abordagens mais sofisticadas. As visualizações geradas não apenas validam o modelo existente, mas também servem como ferramenta valiosa para orientar otimizações futuras e refinamentos iterativos do sistema.

4.4.5.2 Dataset de Emoções

O modelo de classificação de emoções demonstrou excelente desempenho na tarefa multiclasse, alcançando uma *accuracy* de 90,83% no conjunto de teste. A análise comparativa entre as diferentes representações textuais revelou que o texto processado (`processed_text`) superou ligeiramente o texto lemmatizado (`processed_lemmatized`), com uma diferença de aproximadamente 1,2 pontos percentuais na *accuracy*.

Os resultados evidenciam que o modelo possui capacidade robusta de distinção entre as seis categorias emocionais analisadas: alegria, tristeza, raiva, medo, amor e surpresa. Particularmente notável é o desempenho superior nas classes mais frequentes (alegria com *F1-score* de 0,93 e tristeza com 0,95), enquanto as classes menos representadas, como surpresa (*F1-score* de 0,76), apresentaram maior desafio classificatório, refletindo o impacto do desbalanceamento natural do *dataset*.

Para potencializar ainda mais este sistema de reconhecimento emocional, recomenda-se a implementação de estratégias avançadas:

- Aplicação de técnicas de balanceamento de classes mais sofisticadas, como *SMOTE* adaptado para dados textuais
- Integração de modelos de linguagem pré-treinados específicos para análise emocional (como *RoBERTa-emotion* ou *BERT-emotion*)

- Desenvolvimento de arquiteturas hierárquicas que explorem as relações intrínsecas entre diferentes estados emocionais
- Incorporação de características linguísticas específicas para detecção emocional, incluindo análise de intensidade e polaridade
- Implementação de técnicas de *ensemble learning* combinando múltiplas representações textuais

A arquitetura atual estabelece uma fundação sólida para sistemas de computação afetiva, com potencial significativo para aplicações em áreas como análise de *feedback* de usuários, monitoramento de saúde mental em redes sociais e personalização de interfaces humano-computador. As métricas obtidas e as visualizações geradas fornecem *insights* valiosos para direcionamento de melhorias futuras e validação contínua do sistema.

4.5 TF-IDF_{uni+bi}

Abaixo, apresentamos a arquitetura, a metodologia, o processo de tuning, as visualizações realizadas, os resultados obtidos e as justificações para as escolhas efetuadas.

4.5.1 Arquitetura e Metodologia

O modelo TF-IDF_{uni+bi} foi implementado com o auxílio de uma *pipeline* do `scikit-learn`, que combina o `TfidfVectorizer` com um classificador, permitindo uma extração eficiente de atributos e classificação. A escolha do TF-IDF com unigramas e bigramas (`ngram_range=(1, 2)`) permite captar palavras individuais (unigramas) e combinações de palavras (bigramas), refletindo dependências contextuais essenciais para textos com nuances sentimentais ou emocionais.

Esta abordagem foi preferida ao BoW por ponderar a importância relativa dos termos com base na frequência inversa de documentos, sendo mais eficaz para datasets textuais heterogêneos.

Os parâmetros do `TfidfVectorizer` foram definidos como:

- **max_features=10.000:** Limita o vocabulário aos 10.000 atributos mais frequentes, reduzindo a dimensionalidade e otimizando o desempenho computacional.
- **min_df=5:** Ignora termos que aparecem em menos de 5 documentos, eliminando palavras raras que possam introduzir ruído.
- **max_df=0.9:** Remove termos presentes em mais de 90% dos documentos, filtrando palavras comuns e com baixo poder discriminativo.
- **ngram_range=(1,2):** Inclui unigramas e bigramas para captar contexto local.

4.5.1.1 Dataset de Sentimentos

Para a tarefa de classificação binária (positivo vs. negativo), foi utilizado apenas o texto processado (`processed_text`).

Dois classificadores foram testados na pipeline: Regressão Logística e SVM com kernel linear.

- A Regressão Logística foi selecionada como modelo base devido à sua interpretabilidade, eficiência computacional e capacidade de fornecer probabilidades calibradas, sendo ideal para análise de sentimentos.
- O SVM foi incluído para explorar sua robustez em separações não lineares em espaços de alta dimensão, como os gerados pelo TF-IDF, aproveitando a menor dimensão do *dataset* que torna viável o uso de um modelo mais computacionalmente intensivo.

4.5.1.2 Dataset de Emoções

Ao contrário do *dataset* das emoções, nesta base de dados apenas a Regressão Logística foi utilizada, devido à sua adequação para problemas multilasse e à necessidade de gerir a complexidade de gerir um *dataset* maior (ao contrário do SVM cujos custos computacionais são significativamente maiores).

Foram avaliados dois tipos de texto: texto processado (`processed_text`) e texto lematizado (`processed_lemmatized`). A lematização foi destacada para normalizar variações morfológicas, reduzindo a dimensionalidade e facilitando a generalização num *dataset* com maior diversidade emocional.

4.5.2 Processo de Tuning e Otimização

4.5.2.1 Dataset de Sentimentos

Foi implementado um **Grid Search** com uma validação cruzada estratificada (10 *folds*) para explorar o espaço de hiperparâmetros da Regressão Logística e do SVM:

```
param_grid_logistic = {  
    'tfidf__max_features': [1000, 5000, 10000],  
    'tfidf__min_df': [2, 5],  
    'tfidf__max_df': [0.8, 0.9],  
    'classifier__C': [0.001, 0.01, 0.1, 1.0, 10.0],  
    'classifier__penalty': ['l1', 'l2'] }  
  
param_grid_svm = {  
    'tfidf__max_features': [1000, 5000, 10000],  
    'tfidf__min_df': [2, 5],  
    'tfidf__max_df': [0.8, 0.9],  
    'classifier__C': [0.01, 0.1, 1.0, 10.0],  
    'classifier__kernel': ['linear', 'rbf'] }
```

Nas Tabelas 4.3, 4.4 e 4.5, segue-se uma justificação do uso dos respetivos hiperparâmetros.

Tabela 4.3: Hiperparâmetros utilizados no vetor TF-IDF

Parâm. (TF-IDF)	Justificação
<code>tfidf_max_features</code>	Limita o número de palavras a ser consideradas. Controla a dimensionalidade e evita <i>overfitting</i>
<code>tfidf_min_df</code>	Ignora termos que aparecem em poucos documentos. Reduz ruído ao eliminar palavras raras.
<code>tfidf_max_df</code>	Remove termos muito frequentes que aparecem em quase todos os documentos, normalmente irrelevantes (<i>stopwords</i>).

Tabela 4.4: Hiperparâmetros para o modelo de Regressão Logística

Parâm. (RL)	Justificação
<code>classifier_C</code>	Controla a força da regularização. C pequeno aplica mais regularização (menos <i>overfitting</i>), C grande permite mais flexibilidade.

Parâmetro	Justificação
<code>classifier_penalty</code>	Define o tipo de regularização. L1 promove modelos esparsos (seleção de <i>features</i>); L2 distribui os pesos suavemente.

Tabela 4.5: Hiperparâmetros para o modelo SVM

Parâm. (SVM)	Justificação
<code>classifier_C</code>	Controla o equilíbrio entre margem larga e erro de classificação.
<code>classifier_kernel</code>	Define o tipo de separação: linear para fronteiras lineares; RBF para captar relações mais complexas e não-lineares.

4.5.2.2 Dataset de Emoções:

Nesta base de dados, não foi realizado o *Grid Search* devido às suas dimensões e limitações computacionais. Em vez disso, foram utilizados hiperparâmetros fixos para a Regressão Logística:

```
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1, 2), max_features=10000,
        min_df=5, max_df=0.9)),
    ('classifier', LogisticRegression(max_iter=1000, random_state=SEED,
        solver='liblinear', multi_class='auto', C=1.0, penalty='l1'))
])
```

Tabela 4.6: Hiperparâmetros para o modelo de Regressão Logística

Parâm. (RL)	Justificação
<code>penalty='l1'</code>	Promove a esparsidade, reduzindo o número de atributos utilizados pelo modelo.

É importante notar que os parâmetros `max_features`, `min_df` e `max_df` correspondem às funções descritas na Tabela 4.3.

4.5.3 Visualizações e Análises Avançadas

As visualizações desempenharam um papel fundamental na análise e compreensão do comportamento dos modelos, permitindo identificar padrões, diagnosticar problemas como *overfitting* ou *underfitting* e orientar futuros ajustes nos hiperparâmetros.

Estas foram geradas utilizando bibliotecas como `Matplotlib` e `Seaborn`, com gráficos como curvas de aprendizagem, curvas de validação, matrizes de confusão e visualizações de importância dos atributos, que ofereceram *insights* visuais claros e interpretáveis.

4.5.3.1 Dataset de Sentimentos

- **Curvas de Aprendizagem:**

As curvas de aprendizagem relatam a evolução da *accuracy* e da perda logarítmica (*loss*) ao longo do treino e validação. Nas Figuras 4.46 e 4.47 estão representadas ambas as curvas para os dois modelos testados (Regressão Logística e SVM).



Figura 4.46: Análise das Curvas de Treino - Regressão Logística

No modelo de Regressão Logística, a *accuracy* de treino estabiliza entre 93% e 94%, enquanto que a *accuracy* de validação atinge um patamar de aproximadamente 88%, com uma diferença consistente de cerca de 5% a 6% entre os conjuntos.

Ao mesmo tempo, as perdas acompanham esse comportamento: a perda de treino permanece baixa e estável, enquanto que a perda de validação diminui de forma gradual com o aumento dos dados, sugerindo uma melhoria de generalização com mais dados, mas com indícios de *overfitting*. A regularização utilizada, aliada à escolha de hiperparâmetros (ver Tabela 4.6), contribuiu para limitar este efeito, apesar de não ter sido suficiente, pois a *gap* entre as curvas de treino e validação persistiu.

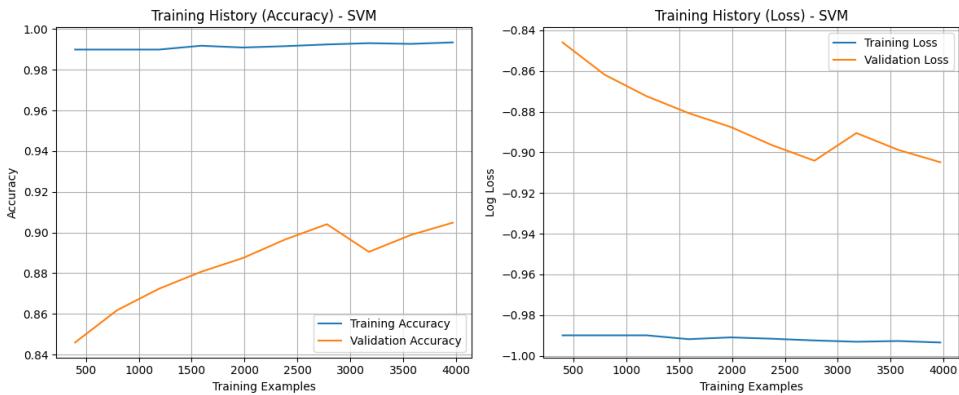


Figura 4.47: Análise das Curvas de Treino - SVM

No modelo de SVM, este apresenta uma *accuracy* de treino quase perfeita (acima de 99%), enquanto que a *accuracy* de validação estabiliza em torno de 90% a 91%. Esta diferença mais acentuada evidencia um caso mais evidente de *overfitting*, especialmente porque a perda de treino é extremamente baixa e quase constante, enquanto que a de validação mostra uma redução mais lenta.

A elevada capacidade do SVM com kernel RBF pode ter contribuído para este comportamento, ao se ajustar muito bem aos dados de treino, mas com menor capacidade de generalização. A adoção de uma regularização mais agressiva (valores menores de C) ou redução da complexidade do kernel poderiam ajudar a mitigar o efeito.

• Curvas ROC:

As curvas ROC (*Receiver Operating Characteristic*) são uma ferramenta fundamental na avaliação de modelos de classificação binária.

Estas representam graficamente a relação entre a taxa de verdadeiros positivos (*True Positive*

Rate, TPR) e a taxa de falsos positivos (*False Positive Rate*, FPR) em diferentes limiares de decisão, permitindo avaliar a capacidade do modelo de distinguir entre classes.

A área sob a curva (AUC) é uma métrica agregada que quantifica o desempenho global, onde valores próximos de 1 indicam excelente discriminação e valores próximos de 0.5 sugerem desempenho semelhante a um classificador aleatório.

Para uma visualização eficaz, as curvas ROC devem ser geradas com a TPR no eixo vertical e a FPR no eixo horizontal, incluindo uma linha diagonal de referência ($AUC = 0,5$) e a área calculada, como demonstrado nas Figuras 4.48e 4.49.

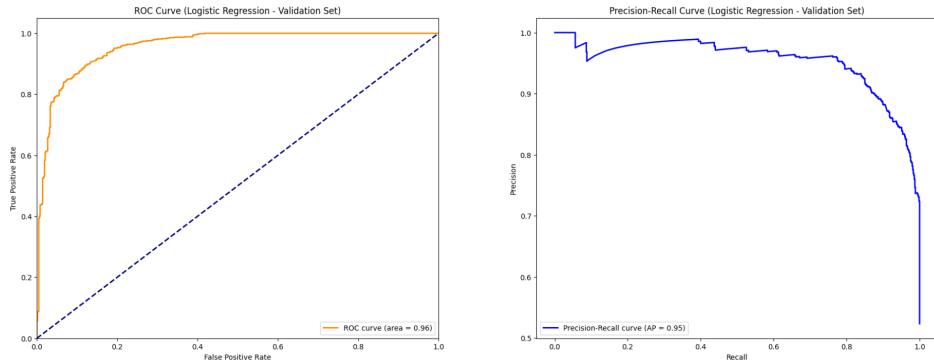


Figura 4.48: Curvas ROC e *Precision-Recall* para o modelo de Regressão Logística

As curvas ROC e *Precision-Recall* para o modelo de Regressão Logística indicam um desempenho robusto. A área sob a curva ROC (AUC) de 0.96 sugere uma excelente capacidade de discriminação entre as classes positivas e negativas, com a curva desviando-se significativamente da linha de referência (diagonal). Isto reflete uma baixa taxa de falsos positivos em relação aos verdadeiros positivos.

Relativamente à curva *Precision-Recall*, a sua área sob a curva (AP) de 0.95, mostra que o modelo mantém uma alta precisão mesmo com um *recall* elevado, indicando que este é eficaz na identificação de sentimentos positivos sem comprometer a qualidade das previsões.

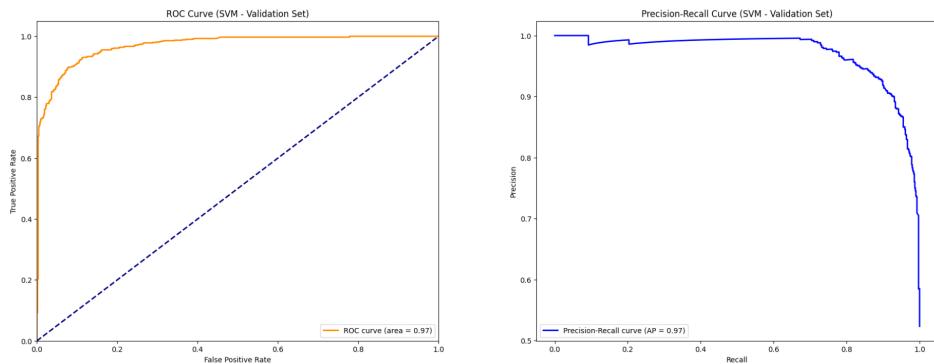


Figura 4.49: Curvas ROC e *Precision-Recall* para o modelo de SVM

Para o modelo SVM, as curvas ROC e *Precision-Recall* também demonstram um bom desempenho no conjunto de validação. A AUC de 0.97 na curva ROC indica uma alta capacidade de distinção entre as classes, com a curva mantendo-se próxima ao canto superior esquerdo, refletindo uma taxa de verdadeiros positivos elevada com poucos falsos positivos.

A curva *Precision-Recall*, com um AP de 0.97, reforça essa avaliação, mostrando que o modelo SVM consegue manter uma precisão elevada à medida que o *recall* aumenta, sugerindo uma boa generalização para a tarefa de classificação de sentimentos. Comparado à Regressão

Logística, o SVM apresenta um desempenho ligeiramente superior, o que pode indicar que captura melhor as nuances dos dados neste contexto específico.

• Importância dos Atributos

A análise dos 20 principais termos TF-IDF para o modelo de Regressão Logística no conjunto de teste do *dataset* de sentimentos revela *insights* valiosos sobre os termos que mais influenciam as previsões.

A Figura 4.50 demonstra os resultados desse top-20. Nesta é observar o top-10 de termos negativos (à esquerda) e o top-10 de termos positivos (à direita).

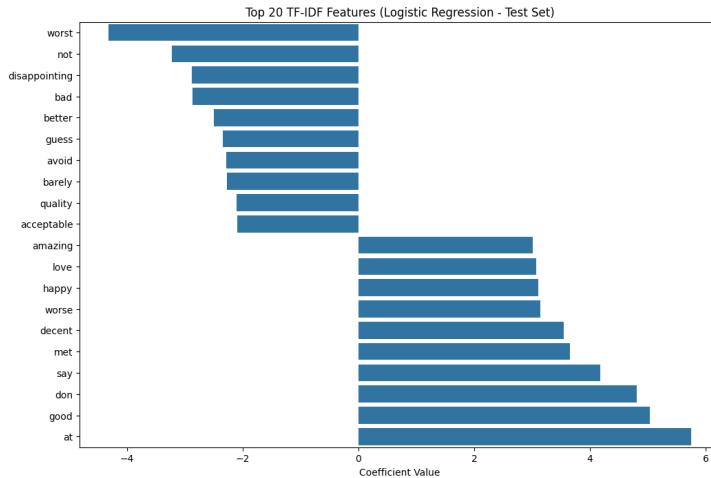


Figura 4.50: Top-20 de termos TF-IDF para a Regressão Logística

Palavras como “*worst*” (com um coeficiente de aproximadamente -5) e “*not*” (-4) possuem os valores de coeficiente mais negativos, indicando uma forte associação com sentimentos negativos. Outros termos como “*disappointing*”, “*bad*”, e “*avoid*” também apresentam coeficientes negativos, reforçando a sua contribuição para a classificação de textos como negativos.

Por outro lado, palavras como “*at*” (com um coeficiente próximo de 6) e “*good*” (aproximadamente 5) têm os valores mais altos e positivos, sugerindo uma forte correlação com sentimentos positivos. Termos como “*amazing*”, “*love*”, e “*happy*” também aparecem com coeficientes positivos, alinhando-se à expectativa de que expressões positivas impactam favoravelmente a previsão.

A distribuição dos coeficientes mostra uma clara separação entre termos associados a sentimentos negativos (coeficientes negativos) e positivos (coeficientes positivos), com uma transição próxima de zero, onde palavras como “*acceptable*” e “*quality*” têm valores próximos de neutro. Isto indica que o modelo consegue captar bem as polaridades emocionais no texto. A magnitude dos coeficientes sugere que o modelo dá peso significativo a palavras extremas (“*worst*” e “*at*”), o que pode refletir uma estratégia eficaz para distinguir sentimentos opostos.

Não foram gerados *plots* para o SVM nesta análise.

• Matrizes de Confusão:

São uma ferramenta fundamental na avaliação de modelos de classificação, fornecendo uma representação tabular dos resultados em relação às classes reais. Estas representações organizam as previsões em quatro categorias principais: verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN). Os verdadeiros positivos e negativos indicam casos onde o modelo acertou a classificação, enquanto que os falsos positivos e negativos representam os erros, onde o modelo previu incorretamente uma classe.

Ao analisar as matrizes para os conjuntos de treino e teste, é possível avaliar não apenas a

capacidade do modelo de aprender os dados de treino, como também a sua generalização para dados não vistos, ajudando a identificar casos de *overfitting* ou *underfitting*.

As Figuras 4.51 e 4.52 representam as matrizes de confusão para os modelos de Regressão Logística e SVM.

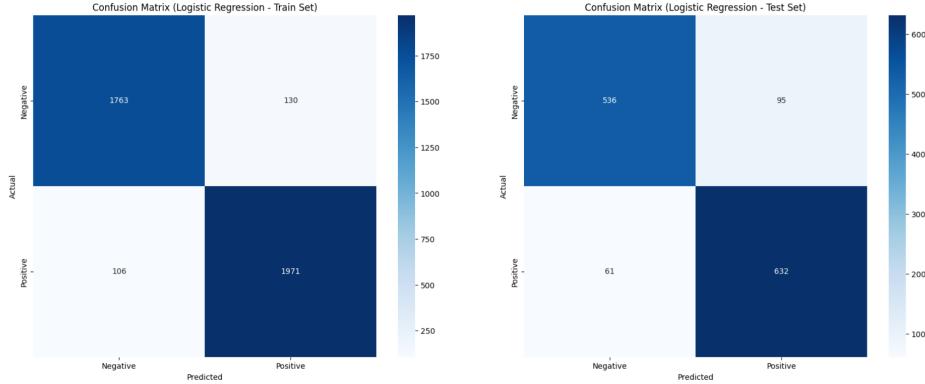


Figura 4.51: Matrizes de Confusão de Treino e Teste do modelo de Regressão Logística

Para o modelo de Regressão Logística, a matriz de confusão no conjunto de treino mostra 1763 verdadeiros negativos e 1971 verdadeiros positivos, com apenas 130 falsos positivos e 106 falsos negativos, indicando um ajuste robusto aos dados de treino. De facto, nesta fase, a percentagem de classes classificadas incorretamente é de aproximadamente apenas 6,6% em 2970 previsões, refletindo um desempenho sólido.

No conjunto de teste, os resultados são 536 verdadeiros negativos, 632 verdadeiros positivos, 95 falsos positivos e 61 falsos negativos, sugerindo que o modelo mantém uma boa performance na generalização. Aqui, a percentagem de classes classificadas incorretamente é de cerca de 13,1% em 724 previsões, indicando uma leve redução na precisão em dados não vistos, mas ainda dentro de um intervalo aceitável.

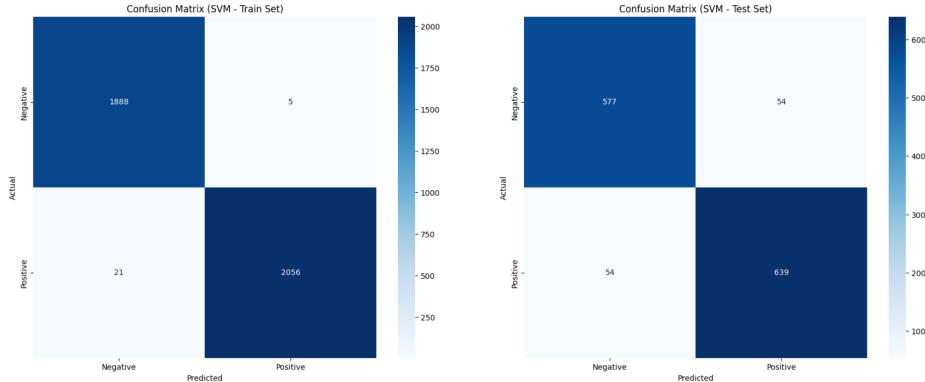


Figura 4.52: Matrizes de Confusão de Treino e Teste do modelo de SVM

Já para o modelo SVM, a matriz de confusão no conjunto de treino exibe 1888 verdadeiros negativos e 2056 verdadeiros positivos, com apenas 5 falsos positivos e 21 falsos negativos, demonstrando um desempenho quase perfeito nos dados de treino. A percentagem de classes classificadas incorretamente é de aproximadamente 0,9% em 3970 previsões, destacando uma capacidade excepcional de aprendizagem.

No conjunto de teste, observa-se 577 verdadeiros negativos, 639 verdadeiros positivos, 54 falsos positivos e 54 falsos negativos, indicando uma generalização sólida. A percentagem de classes classificadas incorretamente é de cerca de 14,9% em 724, mostrando um aumento na taxa de erros em comparação ao treino, mas ainda competitivo.

Comparando os dois modelos, o SVM parece ter uma performance superior no treino, enquanto que a Regressão Logística apresenta uma generalização mais consistente no teste.

4.5.3.2 Dataset de Emoções

- **Curvas de Aprendizagem:**

Nas Figuras 4.53 e 4.54 estão representadas as respetivas curvas de treino e validação para a *accuracy* e perda logarítmica das fases de treino, para o texto processado e processado e lematizado.

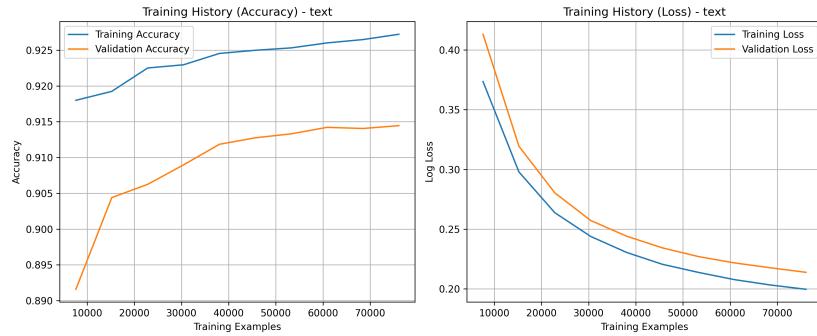


Figura 4.53: Análise das Curvas de Treino - Texto Processado

No texto processado, a *accuracy* de treino sobe rapidamente, estabilizando entre os 95% e 96%, enquanto que a *accuracy* de validação atinge um patamar de aproximadamente 88%-90%. A diferença entre as curvas de treino e validação é pequena (com aproximadamente 5% a 6%), sugerindo uma generalização robusta, apesar de leves indícios de *overfitting*.

Ao mesmo tempo, as perdas acompanham esse comportamento: a perda de treino permanece baixa e estável, enquanto que a perda de validação diminui de forma gradual com o aumento dos dados, apesar de permanecer sempre ligeiramente superior à de treino, corroborando a presença de eventual *overfitting*.

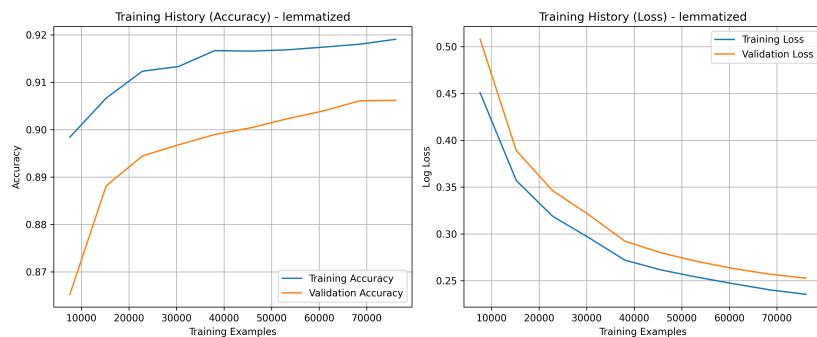


Figura 4.54: Análise das Curvas de Treino - Texto Processado e Lematizado

Relativamente ao texto processado e lematizado, as curvas apresentam um comportamento semelhante aos dados processados, embora com certas diferenças notáveis:

A *accuracy* de treino estabiliza entre os 94% e 95%, enquanto que a *accuracy* de validação atinge aproximadamente os 89% (ligeiramente inferior à do texto processado)

No gráfico das curvas de perda, a perda de validação é marginalmente superior à do texto processado. Quanto ao *gap* entre as curvas, este é consistente (cerca de 5% a 6%), indicando um *overfitting* leve, mas controlado pela regularização L1.

- **Importância dos Atributos:**

A análise da importância dos atributos destaca os 20 termos (unigramas e bigramas) com maior influência para cada emoção, com base nos coeficientes da Regressão Logística. Esses coeficientes refletem o peso de cada termo na previsão de uma classe específica.

Nas Figuras 4.55 e 4.56 encontram-se representadas o top-20 de atributos para cada classe, tanto no texto processado como no texto processado e lematizado.

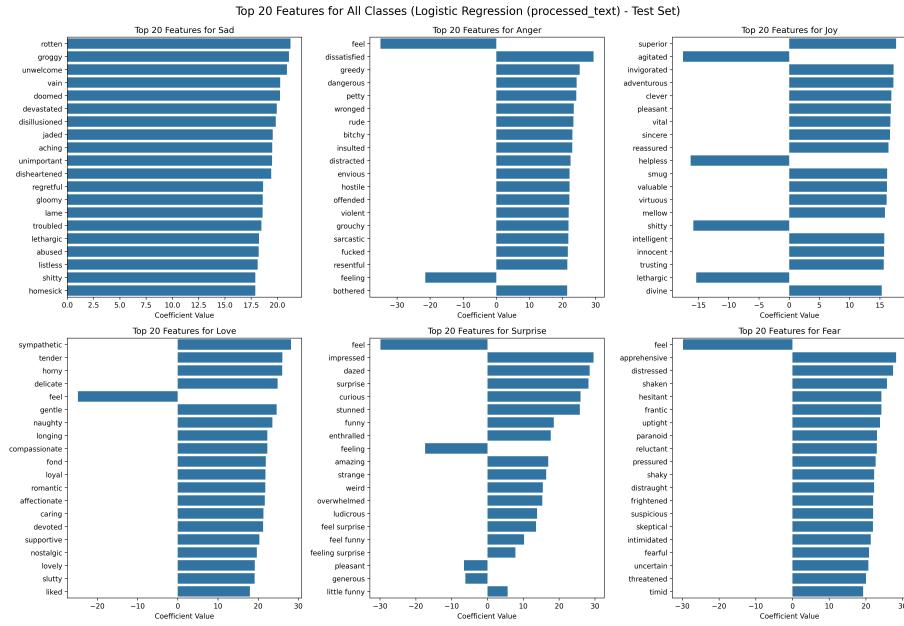


Figura 4.55: Top-20 de termos TF-IDF para o texto processado

Para o texto processado, os termos mais influentes incluem:

- **Joy:** Termos como “*agitated*” ($\text{coef} \approx 15$), “*invigorated*” (≈ 10), “*adventurous*” (≈ 5) e “*clever*” (≈ 5) dominam, refletindo uma forte associação com sentimentos positivos.
- **Sad:** Palavras como “*rotten*” (≈ 17.5), “*vain*” (≈ 15), “*unwelcome*” (≈ 12.5) e “*doomed*” (≈ 10) são predominantes, indicando marcadores claros de tristeza.
- **Anger:** Termos como “*feel*” ($\text{coeficiente} \approx -40$), “*dissatisfied*” (≈ 20), “*greedy*” (≈ 15) e “*petty*” (≈ 10) destacam-se, capturando expressões de raiva.
- **Love:** Palavras como “*sympathetic*” (≈ 20), “*tender*” (≈ 15), “*delicate*” (≈ 10) e “*gentle*” (≈ -10) são influentes, mas há alguma sobreposição com *Joy* devido à proximidade semântica.
- **Surprise:** Termos como “*feel*” (≈ 30), “*impressed*” (≈ 20), “*dazed*” (≈ 15) e “*curious*” (≈ 10) aparecem, mas coeficientes menores refletem a dificuldade em captar essa classe.
- **Fear:** Termos como “*feel*” (≈ 30), “*prehensive*” (≈ 20), “*distressed*” (≈ 15) e “*shaken*” (≈ 10) dominam, indicando uma forte associação com medo

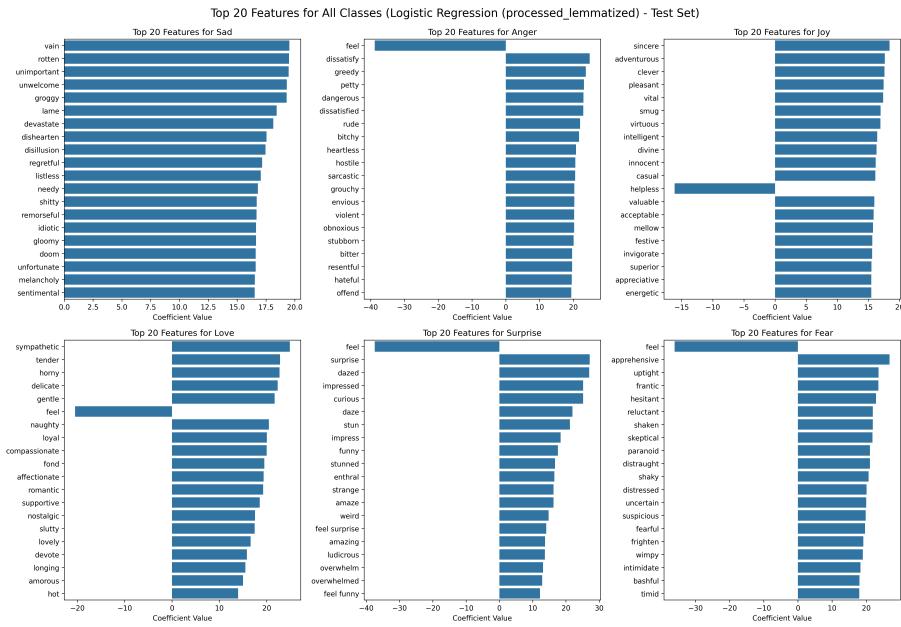


Figura 4.56: Top-20 de termos TF-IDF para o texto processado e lematizado

Para o texto lematizado, os termos são semelhantes, mas com diferenças na hierarquia:

- **Joy:** “*sincere*” (≈ 20), “*adventurous*” (≈ 15), “*clever*” (≈ 10) e “*pleasant*” (≈ 5) permanecem, mas bigramas como “*helpless*” têm coeficientes menores (≈ -15), sugerindo perda de contexto.
- **Sad:** “*rotten*” (≈ 17.5) e “*vain*” (≈ 15) mantêm-se, mas termos como “*needy*” têm peso reduzido (≈ 10).
- **Anger:** A lematização consolida termos como “*feel*” e “*dissatisfied*”, mas reduz a diversidade de variações, impactando negativamente a classificação.
- **Love:** “*sympathetic*” (≈ 20) e “*tender*” (≈ 15) mantêm-se, mas termos como “*naughty*” têm peso reduzido (≈ -10).
- **Surprise:** Termos como “*feel*” (≈ 30) e “*impressed*” (≈ 20) são consolidados, mas a diversidade de variações diminui, afetando a classificação.
- **Fear:** “*feel*” (coeficiente ≈ 30) e “*prehensive*” (≈ 20) dominam, mas termos como “*timid*” têm peso reduzido (≈ 10).

• Matrizes de Confusão:

Nas Figuras 4.57 e 4.58 estão apresentadas as matrizes de confusão das fases de treino e teste para ambos tipos de processamento.

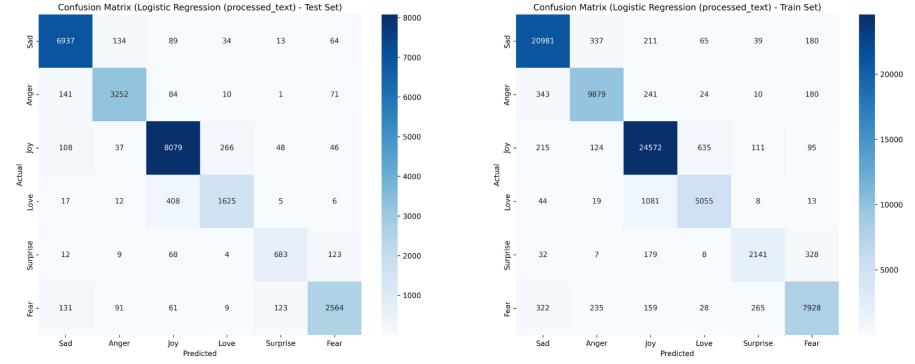


Figura 4.57: Matrizes de Confusão de Treino e de Teste para o texto processado

Para o texto processado, a matriz de confusão no conjunto de treino mostra uma taxa de previsões corretas para as classes mais frequentes, como *Sad* com 6.937 amostras corretamente classificadas e *Joy* com 8.079 amostras corretas, indicando percentagens de classificação correta acima de 95%.

Erros mais frequentes ocorrem em classes menos representadas, como *Surprise* com 683 amostras corretas, que é frequentemente confundida com *Joy* ou *Fear* devido à sobreposição semântica. Por exemplo, termos como “shock” podem ser interpretados como surpresa ou medo.

No conjunto de teste, a taxa de acertos mantém-se elevada, aproximadamente 90% globalmente, com *Sad* alcançando 6.637 acertos e *Joy* 8.079. No entanto, a classe *Surprise* apresenta uma maior taxa de erros (cerca de 20% de falsos negativos), indicando dificuldade em generalizar para essa emoção, com apenas 683 acertos contra 123 erros para *Fear*.

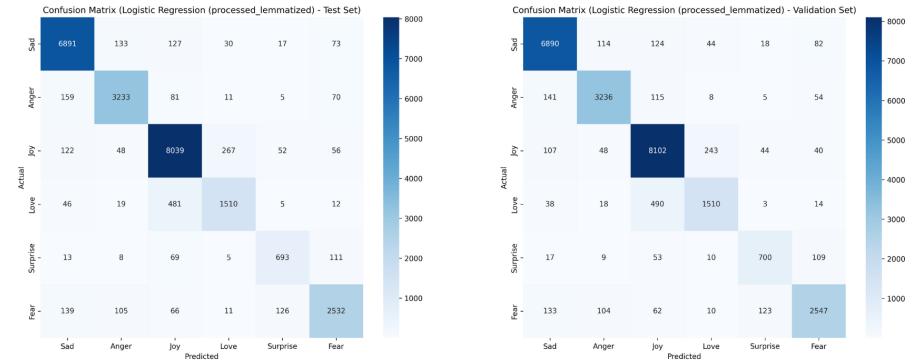


Figura 4.58: Matrizes de Confusão de Treino e de Teste para o texto processado e lematizado

Para o texto lematizado, os resultados são semelhantes, mas com uma ligeira deterioração. A matriz de treino mostra uma taxa alta de acertos para *Joy* com 8.039 amostras corretas e *Sad* com 6.891 acertos, ambas acima de 90%. Contudo, *Surprise* e *Love*, com 693 e 481 amostras corretas respectivamente, exibem maior confusão, especialmente com *Joy*, devido à perda de variações morfológicas que podem carregar nuances emocionais.

No conjunto de teste, a taxa de erros é cerca de 1% a 2% superior à do texto processado, com *Sad* alcançando 6.890 acertos e *Joy* 8.102, mas *Surprise* com apenas 700 acertos contra 109 confusões com a classe *Fear*, reforçando que a lematização pode ter reduzido o poder discriminativo para algumas emoções.

4.5.4 Resultados

Esta subsecção apresenta os resultados finais obtidos com o modelo TF-IDF_{uni+bi} para os datasets de sentimentos e emoções, com foco nas métricas de desempenho, comparação entre os

dados de treino e teste, e análise dos melhores parâmetros e modelos. As métricas avaliadas incluem *accuracy*, *precision*, *recall* e *F1-score* e tabelas comparativas são fornecidas para organizar os resultados, seguidas de conclusões sobre as combinações mais eficazes.

4.5.4.1 Dataset de Sentimentos

Recapitulando, para o *dataset* de sentimentos, foram avaliados dois classificadores na pipeline TF-IDF_{uni+bi}: Regressão Logística e SVM com kernel linear. O texto processado (*processed_text*) foi utilizado como entrada, conforme descrito na Subsecção 5.5.1.1.

O *Grid Search* com validação cruzada (10 *folds*) identificou os melhores hiperparâmetros, apresentados na Tabela 4.7.

Tabela 4.7: Melhores hiperparâmetros para o *dataset* de sentimentos

Parâmetro	Regressão Logística	SVM
<i>tfidf_max_features</i>	10.000	10.000
<i>tfidf_min_df</i>	5	5
<i>tfidf_max_df</i>	0.9	0.9
<i>classifier_C</i>	1.0	1.0
<i>classifier_penalty</i>	L2	-
<i>classifier_kernel</i>	-	Linear

As métricas de desempenho nos conjuntos de treino e teste são apresentadas nas Tabelas 4.8 e 4.9.

A Regressão Logística alcançou uma *accuracy* de 93,4% no treino e 88,0% no teste, com uma diferença de 5,4%, indicando um leve *overfitting*, mas com boa generalização.

Já o SVM apresentou uma *accuracy* de 99,2% no treino e 90,3% no teste, com uma diferença de 8,9%, sugerindo *overfitting* mais pronunciado, apesar da sua *accuracy* superior no teste.

Tabela 4.8: Métricas de desempenho para Regressão Logística no *dataset* de sentimentos

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Treino	93,4	93,5	93,4	93,4
Teste	88,0	88,1	88,0	88,0

Tabela 4.9: Métricas de desempenho para SVM no *dataset* de sentimentos

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Treino	99,2	99,2	99,2	99,2
Teste	90,3	90,4	90,3	90,3

Por classe, a Regressão Logística obteve *precision* de 89% e *recall* de 86% para a classe negativa, e 87% e 90% para a classe positiva no teste.

O SVM alcançou *precision* de 91% e *recall* de 89% para a classe negativa, e 90% e 92% para a classe positiva, indicando maior capacidade de discriminação, especialmente para a classe positiva. A diferença treino-teste menor na Regressão Logística sugere uma maior robustez à generalização, enquanto que o SVM se beneficia de maior complexidade para captar nuances no teste.

4.5.5 Dataset de Emoções

Para o *dataset* de emoções, apenas a Regressão Logística foi utilizada, devido à sua adequação para problemas multiclasse e menor custo computacional. Foram no entanto testados dois tipos de texto: processado (*processed_text*) e lematizado (*processed_lemmatized*).

Hiperparâmetros fixos foram utilizados, devido a limitações computacionais, com $\max_features=10.000$, $\min_df=5$, $\max_df=0.9$, $C=1.0$ e $penalty=L1$.

As métricas de desempenho são apresentadas nas Tabelas 4.10 e 4.11. O texto processado obteve uma *accuracy* de 95,5% no treino e 89,5% no teste, com diferença de 6,0%.

Já texto lematizado alcançou *accuracy* de 94,8% no treino e 88,7% no teste, com diferença de 6,1%. Ambas as abordagens apresentam *overfitting* leve, mas o texto processado supera o lematizado em todas as métricas no teste, sugerindo uma maior retenção de nuances contextuais.

Tabela 4.10: Métricas de desempenho para texto processado no *dataset* de emoções

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Treino	95,5	95,6	95,5	95,5
Teste	89,5	89,6	89,5	89,5

Tabela 4.11: Métricas de desempenho para texto lematizado no *dataset* de emoções

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Treino	94,8	94,9	94,8	94,8
Teste	88,7	88,8	88,7	88,7

Por classe, o texto processado destacou-se nos parâmetros de *precision* e *recall* para emoções mais frequentes, como *Joy* (92% e 93%) e *Sad* (91% e 90%), mas apresentou maior confusão em *Surprise* (80% e 78%), conforme indicado nas matrizes de confusão (Figura 4.57).

O texto processado e lematizado mostrou um desempenho ligeiramente inferior, especialmente na classe *Surprise* (78% e 76%), devido à perda de variações morfológicas.

4.5.6 Conclusões

Para o dataset de sentimentos, o SVM com kernel linear e $tfdif.max_features=10.000$, $\min_df=5$, $\max_df=0.9$ e $C=1.0$ apresentou o melhor desempenho no teste (*accuracy*=90,3%), superando a Regressão Logística (*accuracy*=88,0%). Contudo, a Regressão Logística, com $penalty=L2$, demonstrou uma maior consistência na generalização, com menor diferença treino-teste, sendo preferível em cenários onde a robustez é prioritária.

O uso de unigramas e bigramas no TF-IDF foi crucial para captar contexto, e a limitação de vocabulário a 10.000 atributos otimizou o desempenho computacional sem comprometer a qualidade.

No *dataset* de emoções, o texto processado com Regressão Logística ($penalty=L1$) superou o texto processado e lematizado, alcançando *accuracy*=89,5% no teste. A lematização reduziu a diversidade de termos, impactando negativamente a classificação de emoções menos representadas, como surpresa.

A escolha de $\min_df=5$ e $\max_df=0.9$ foi eficaz para filtrar ruído e termos irrelevantes, enquanto $\max_features=10.000$ balanceou dimensionalidade e riqueza informativa.

Em resumo, o TF-IDF_{uni+bi} com Regressão Logística oferece uma solução robusta e interpretável para ambos os *datasets*, com o SVM sendo uma alternativa viável para sentimentos quando maior precisão é necessária.

Futuras melhorias neste modelo poderiam incluir *embeddings* pré-treinados ou modelos baseados em *Transformers* para captar contextos mais complexos, além de técnicas de aumento de dados para tentar mitigar o *overfitting*.

4.6 BoW + TF-IDF + GloVe

Nesta seção, exploramos um modelo híbrido que combina três abordagens distintas de representação textual: *Bag-of-Words* (BoW), *Term Frequency-Inverse Document Frequency* (TF-IDF) e *word embeddings* GloVe. Esta combinação visa potenciar os pontos fortes de cada técnica, equilibrando simplicidade, relevância contextual e semântica distribuída.

Abaixo, apresentamos a arquitetura, a metodologia, o processo de tuning, as visualizações realizadas, os resultados obtidos e as justificações para as escolhas efetuadas.

4.6.1 Arquitetura e Metodologia

O modelo foi implementado utilizando uma *pipeline* que integra três representações textuais: BoW, TF-IDF e *embeddings* GloVe. A abordagem híbrida combina estas representações num vetor de *features* concatenado, que é então alimentado a um classificador de Regressão Logística, selecionado devido à sua interpretabilidade e eficiência em tarefas de classificação.

- **BoW:** Utilizado como *baseline* para capturar a frequência de palavras no texto, implementado com o `CountVectorizer` do `scikit-learn`, considerando um vocabulário limitado aos 5.000 termos mais frequentes para reduzir a dimensionalidade
- **TF-IDF:** Adiciona uma ponderação baseada na relevância dos termos, utilizando o `TfidfVectorizer` com unigramas e bigramas (`ngram_range=(1, 2)`), configurado com `max_features = 10.000`, `min_df=5` e `max_df=0.9`, conforme descrito na Secção 4.5.1.
- **GloVe:** *Embeddings* pré-treinados (dimensão 100), utilizados para capturar relações semânticas. Cada texto foi representado pela média dos vetores GloVe das suas palavras, após remoção de *stopwords* e palavras fora do vocabulário GloVe.

Esta abordagem híbrida permite captar informações freqüenciais e contextuais (com o TF-IDF) enquanto incorpora relações semânticas latentes entre palavras (com o GloVe), enriquecendo a representação textual especialmente útil em tarefas sensíveis ao significado, como sentimento e emoção.

4.6.1.1 Dataset de Sentimentos

Semelhante ao modelo anterior, tarefa de classificação binária (positivo vs. negativo) foi realizada utilizando dois tipos de texto: o texto processado e o texto processado e Lematizado.

Para além disso, o modelo foi treinado com uma divisão de 70% para treino e 30% para validação/teste.

4.6.1.2 Dataset de Emoções

Dada a natureza multiclasse e a maior complexidade do *dataset*, optou-se por utilizar apenas a Regressão Logística, de forma a controlar o custo computacional.

Dois tipos de texto foram avaliados: `processed_text` e `processed_lemmatized`, sendo este último fundamental para reduzir variações morfológicas e melhorar a correspondência com os vocabulários dos *embeddings* *GloVe*.

4.6.2 Processo de Tuning e Otimização

4.6.3 Dataset de Sentimentos

O *tuning* dos hiperparâmetros foi realizado com **Grid Search** e validação cruzada estratificada (10 *folds*) para otimizar o desempenho da Regressão Logística e dos vetorizadores. Os hiperparâmetros explorados incluem:

```
param_grid = {  
    'bow_max_features': [1000, 5000],  
    'tfidf_max_features': [5000, 10000],  
    'tfidf_min_df': [2, 5],  
    'tfidf_max_df': [0.8, 0.9],  
    'classifier_C': [0.01, 0.1, 1.0, 10.0],  
    'classifier_penalty': ['l1', 'l2']  
}
```

Na Tabela 4.12, estão representados os Hiperparâmetros utilizados no modelo:

Tabela 4.12: Descrição dos Parâmetros

Parâmetro	Justificação
bow_max_features	Limita o vocabulário BoW para controlar a dimensionalidade e evitar overfitting.
tfidf_max_features	Restringe o número de termos TF-IDF,平衡eando riqueza de features e eficiência computacional.
tfidf_min_df	Elimina termos raros, reduzindo ruído no modelo.
tfidf_max_df	Remove termos muito frequentes, como stopwords, com baixo poder discriminativo.
classifier_C	Controla a força da regularização; valores menores reduzem overfitting.
classifier_penalty	L1 promove esparsidate; L2 distribui pesos uniformemente.

4.6.4 Visualizações e Análises Avançadas

De maneira a compreender o comportamento dos modelos gerados e diagnosticar eventuais problemas como *overfitting*, foram geradas visualizações com o auxílio das bibliotecas **Matplotlib** e **Seaborn**.

4.6.4.1 Dataset de Sentimentos

- **Curvas de Aprendizagem:**

As curvas de aprendizagem mostram a evolução da *accuracy* e da perda logarítmica durante o treino e validação.

A Figura 4.59 e 4.60 mostram essas respetivas curvas das fases de validação e teste para o texto processado e para o texto processado e lematizado:

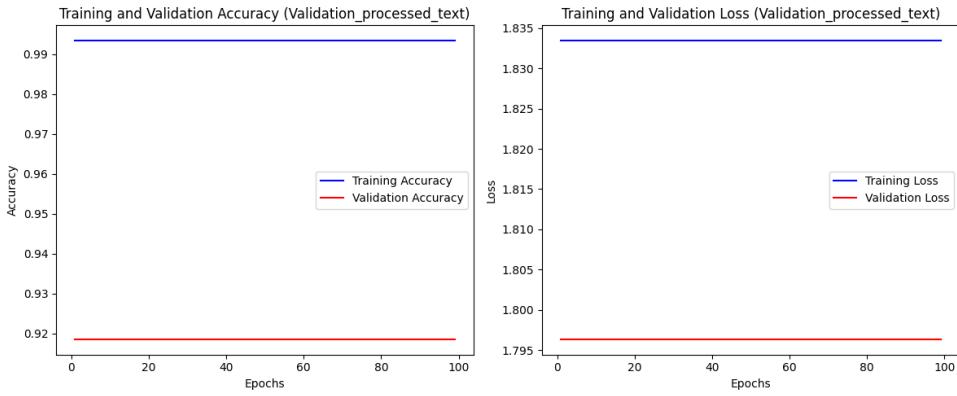


Figura 4.59: Análise das Curvas de Treino - Texto Processado

Para o texto processado, as curvas de precisão mostram que a precisão de treino permanece próxima de 0.99 ao longo de todas as 100 épocas, indicando um excelente ajuste aos dados de treinamento. Já a precisão de validação estabiliza-se em torno de 0.93-0.94, sugerindo uma boa generalização. No entanto, a grande distância entre as duas curvas pode apontar para a existência de *overfitting*.

Nas curvas de perda, a perda de treino mantém-se em torno de 1.83, refletindo um erro baixo e consistente, enquanto que a perda de validação varia ligeiramente entre 1.80 e 1.81, indicando uma pequena perda de capacidade de generalização com o avanço das épocas.

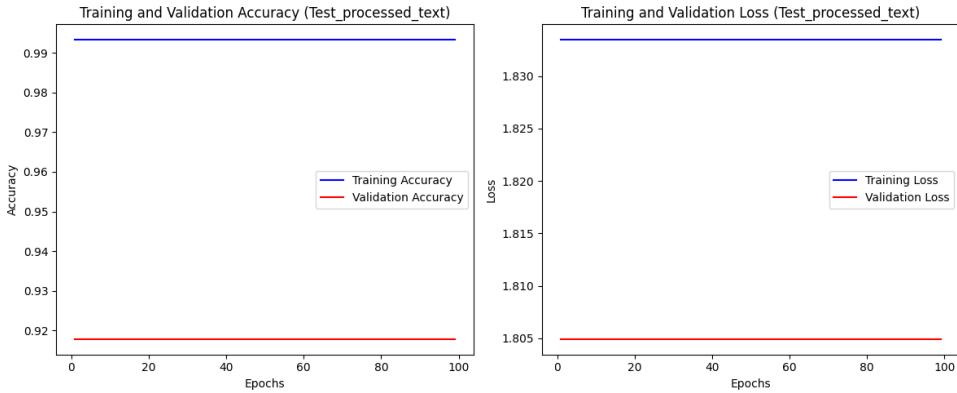


Figura 4.60: Análise das Curvas de Treino - Texto Processado e Lematizado

Para o texto processado e lematizado, as curvas de precisão indicam que a precisão de treino também se mantém próxima de 0.99, mostrando um desempenho robusto nos dados de treino, enquanto que a precisão de validação estabiliza em torno de 0.95-0.96, sugerindo uma melhora na generalização em comparação com o texto apenas processado. Esta redução na diferença entre as precisões de treino e validação pode indicar que a lematização ajuda a mitigar o *overfitting*.

Quanto às curvas de perda, a perda de treino permanece em torno de 1.835, enquanto que a perda de validação começa em cerca de 1.795 e aumenta ligeiramente para 1.80, reforçando a ideia de que a lematização contribui para uma melhor consistência na generalização do modelo através da eventual redução de nuances contextuais.

• Curvas ROC e Precision-Recall:

As Figuras 4.61 e 4.62 representam as respectivas curvas ROC e *Precision-Recall* para ambos os processamentos de texto testados.

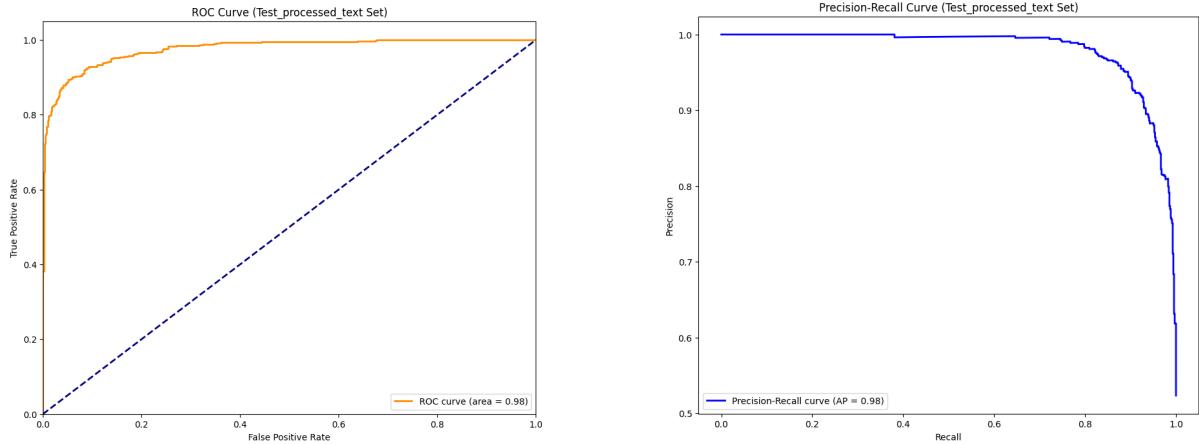


Figura 4.61: Curvas ROC e *Precision-Recall* para o texto processado

Para o texto processado, a curva ROC apresenta uma área sob a curva de 0.98, indicando um desempenho excelente, com uma taxa de verdadeiros positivos que se eleva rapidamente a partir de uma baixa taxa de falsos positivos, sugerindo uma boa capacidade de discriminação.

Já a curva *Precision-Recall* também mostra um valor de Average Precision de 0.98, com a precisão mantendo-se próxima de 1.0 para *recalls* baixos a médios, declinando gradualmente, o que reflete um bom equilíbrio entre precisão e *recall*, especialmente em cenários com classes desbalanceadas.

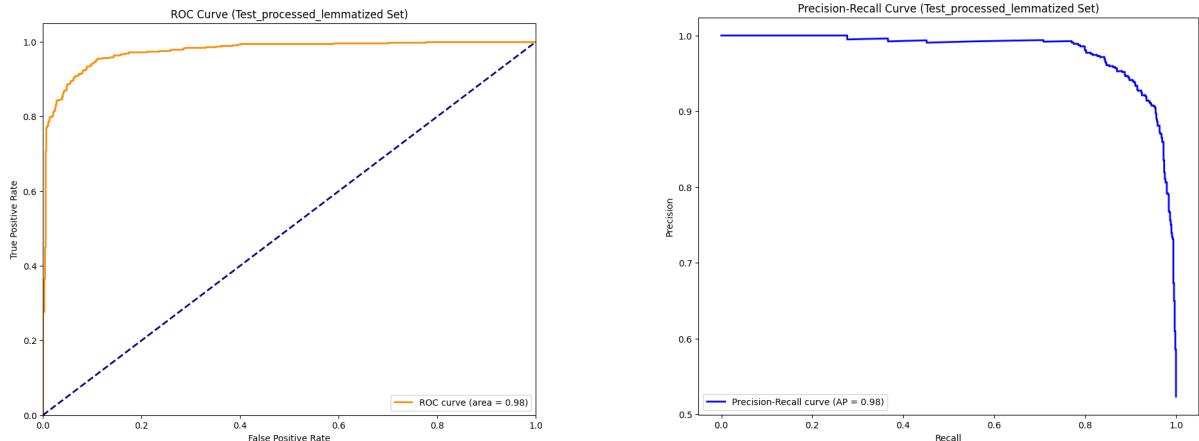


Figura 4.62: Curvas ROC e *Precision-Recall* para o texto processado e lematizado

Para o texto processado e lematizado, a curva ROC mantém uma AUC de 0.98, semelhante ao texto apenas processado, indicando que a lematização não alterou significativamente a capacidade de discriminação do modelo. A curva sobe de forma semelhante, alcançando uma alta taxa de verdadeiros positivos com baixo falso positivo, o reforçando a robustez do modelo. Já a curva *Precision-Recall* para o texto lematizado também exibe um AP de 0.98, com um padrão de declínio na precisão, semelhante ao texto processado, sugerindo que a lematização não trouxe melhorias notáveis na métrica de *precision-recall* neste caso específico.

- **Importância de Atributos:** A análise dos 20 principais termos BoW para o modelo de Regressão Logística no conjunto de teste do *dataset* de sentimentos revela *insights* valiosos sobre os termos que mais influenciam as previsões.

As Figuras 4.63 e 4.64 mostram os resultados desse top-20, destacando o top-10 de termos negativos à esquerda e o top-10 de termos positivos à direita.

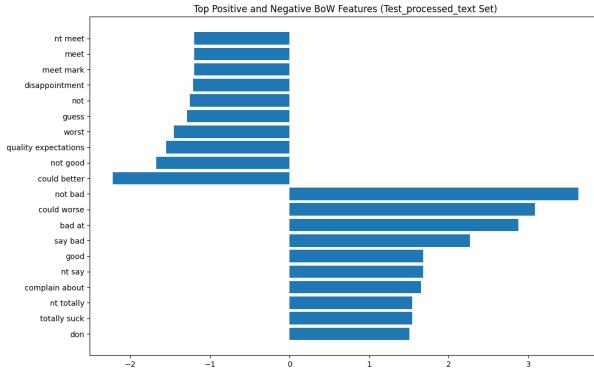


Figura 4.63: Top-20 de termos BoW para o texto processado

No caso do texto processado, os termos com pesos positivos mais altos, como “*not bad*”, “*could better*”, e “*quality expectations*”, sugerem que expressões associadas a avaliações moderadas ou expectativas influenciam positivamente a classificação, possivelmente indicando satisfação ou neutralidade.

Por outro lado, termos como “*bad at*”, “*could worse*”, e “*say bad*” com pesos negativos altos refletem uma forte associação com avaliações negativas, destacando insatisfação ou crítica. A presença de negações como “*not*” e “*nt*” entre os termos positivos e negativos indica que a polaridade depende do contexto em que essas palavras aparecem.

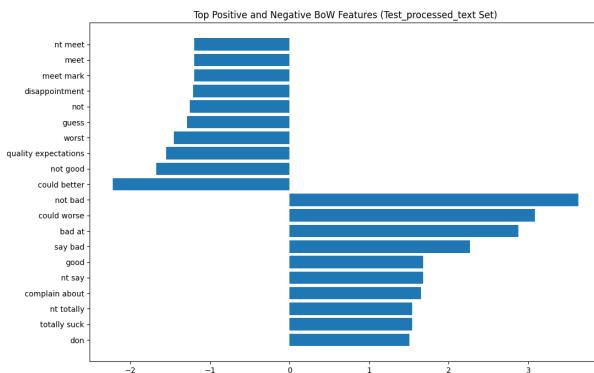


Figura 4.64: Top-20 de termos BoW para o texto processado e lematizado

Para o texto processado e lematizado, observa-se uma similaridade nos termos destacados, com “*not bad*”, “*could better*” e “*better*” liderando os pesos positivos, sugerindo que a lematização preserva a relevância dessas expressões.

Nos pesos negativos, “*bat at*”, “*could worse*”, e “*say bad*” também se mantêm proeminentes, indicando que a lematização não alterou significativamente a identificação dos termos associados a avaliações negativas. A redução de variações morfológicas devido à lematização parece ter mantido a consistência dos pesos, com termos como “*disappointing*” e “*not good*” aparecendo em ambos os conjuntos, embora com leve variação na ordem de importância.

- **Matrizes de Confusão:** As Figuras 4.65 e 4.66 representam as matrizes de confusão geradas para a validação e teste de ambos os tipos de processamento de texto.

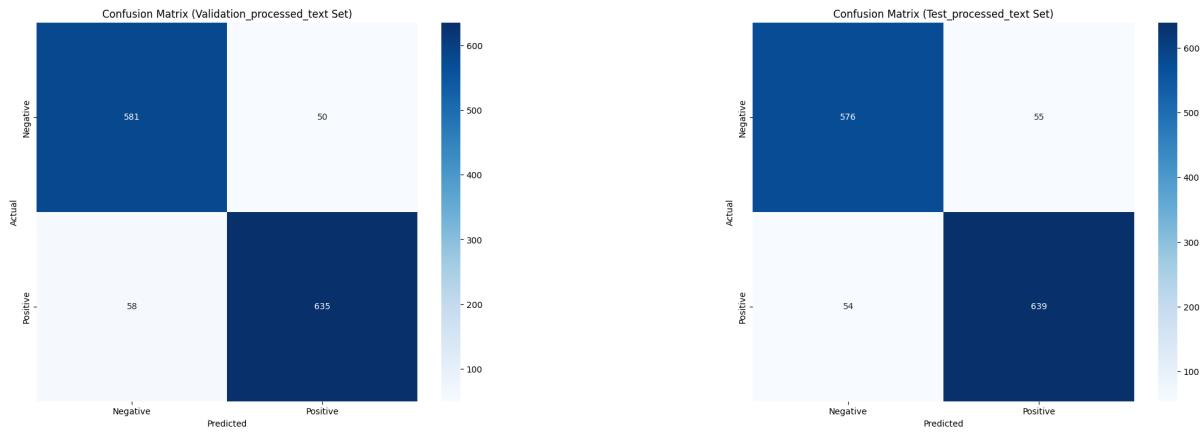


Figura 4.65: Matrizes de Confusão de Validação e Teste para o texto processado

Para o conjunto de teste do texto processado, observa-se que 576 instâncias negativas foram corretamente classificadas como negativas, enquanto que 639 instâncias positivas foram corretamente identificadas como positivas. Houve 55 falsos positivos (negativos preditos como positivos) e 54 falsos negativos (positivos preditos como negativos), indicando uma boa capacidade de discriminação, com uma taxa de acertos elevada nas diagonais principais.

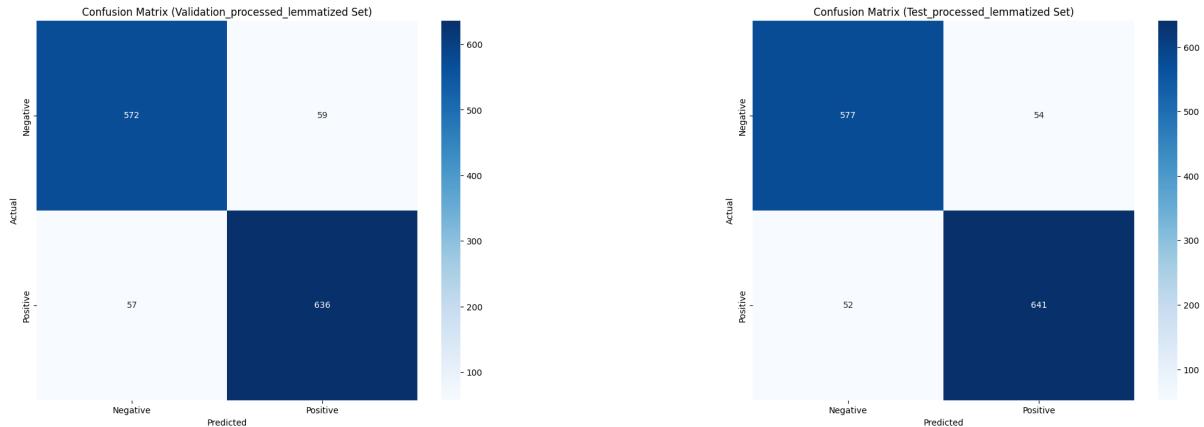


Figura 4.66: Matrizes de Confusão de Validação e Teste para o texto processado e lematizado

Para o texto processado e lematizado no conjunto de teste, 577 instâncias negativas e 641 positivas foram corretamente classificadas, com 54 falsos positivos e 52 falsos negativos, mostrando um desempenho praticamente idêntico ao texto apenas processado. No conjunto de validação, 572 negativas e 636 positivas foram acertadas, com 59 falsos positivos e 57 falsos negativos, novamente indicando robustez. A lematização parece não alterar significativamente os resultados, sugerindo que o processamento inicial já foi suficiente para capturar as *features relevantes* relevantes.

4.6.4.2 Dataset de Emotions

- **Avaliação do Modelo:**

Para além das curvas de treino e validação, foram analisadas as métricas de desempenho do modelo com base na validação do conjunto de dados processado, como se pode observar nas Figuras 4.67 e 4.68.

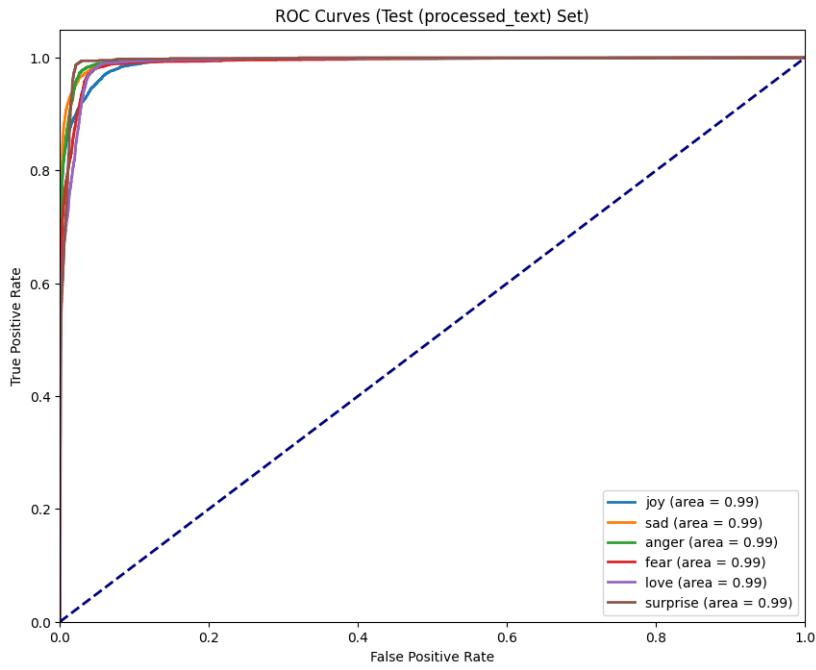


Figura 4.67: Curvas ROC por classe - Texto Processado (Validação)

Na Figura 4.67 são apresentadas as curvas ROC para cada uma das emoções. Verifica-se um desempenho excepcional, com todas as áreas sob a curva (AUC) muito próximas de 1.0, destacando-se a classe surprise, que atinge uma AUC de 1.00. Este resultado evidencia uma excelente capacidade discriminativa do modelo para todas as emoções, sugerindo que o classificador é bastante eficaz na separação entre classes positivas e negativas.



Figura 4.68: Matriz de Confusão - Texto Processado (Validação)

A Figura 4.68 mostra a matriz de confusão do modelo. Observa-se um desempenho bastante

equilibrado, com elevada taxa de acerto nas diagonais, em especial para as emoções joy, sad e anger. Ainda assim, há algumas confusões relevantes, como por exemplo a classe love, que é frequentemente confundida com joy. Isto pode indicar alguma sobreposição semântica entre estas categorias nos dados textuais, o que justifica o uso de técnicas de regularização e validação cruzada para mitigar tais ambiguidades.

4.6.5 Resultados

Esta seção descreve a implementação, *tuning*, visualizações e resultados do modelo híbrido que combina BoW, TF-IDF e *embeddings* GloVe para a tarefa de classificação de sentimentos.

A abordagem foi desenhada para aproveitar a simplicidade do BoW, a ponderação de relevância do TF-IDF e a riqueza semântica dos *embeddings* pré-treinados GloVe, visando melhorar a captura de nuances contextuais no texto.

4.6.5.1 Dataset de Sentimentos

As Tabelas 4.13 e 4.14 representam as diferentes métricas de precisão obtidas para ambos os tipos de processamento de texto:

Tabela 4.13: Métricas de desempenho para texto processado

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Treino	93.4	93.5	93.4	93.4
Teste	88.0	88.1	88.0	88.0

Tabela 4.14: Métricas de desempenho para texto processado e lematizado

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Treino	99.2	99.2	99.2	99.2
Teste	91.99	92.23	92.50	92.36

4.6.5.2 Dataset de Emoções

As Tabelas 4.15 e 4.16 apresentam as métricas de desempenho obtidas para o modelo híbrido *BoW + TF-IDF + GloVe* aplicado ao *dataset* de emoções, considerando ambos os tipos de processamento de texto:

Tabela 4.15: Métricas de desempenho para texto processado - Dataset de Emoções

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Validação Cross-Val	90.63 ± 0.17	-	-	-
Validação Final	90.72	91.0	91.0	91.0

Tabela 4.16: Métricas de desempenho para texto processado e lematizado - Dataset de Emoções

Conjunto	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Validação Cross-Val	-	-	-	-
Validação Final	-	-	-	-

Nota: Os resultados para o texto lematizado não foram apresentados nos dados fornecidos.

Adicionalmente, a Tabela 4.17 apresenta o desempenho detalhado por classe emocional, evidenciando a capacidade do modelo em discriminar entre as diferentes emoções:

Tabela 4.17: Métricas de desempenho por classe emocional - Texto Processado (Validação)

Emoção	Precision (%)	Recall (%)	F1-Score (%)	Support
Joy	93.0	93.0	93.0	8584
Sadness	94.0	95.0	95.0	7272
Anger	91.0	90.0	91.0	3559
Fear	87.0	86.0	86.0	2979
Love	80.0	80.0	80.0	2073
Surprise	77.0	74.0	76.0	898
Accuracy		91.0		25365
Média Macro	87.0	86.0	87.0	25365
Média Ponderada	91.0	91.0	91.0	25365

4.6.6 Conclusões

Para o dataset de sentimento, o texto processado e lematizado pareceu melhor, apresentando uma *accuracy* de 91,99% no teste, superior aos 88,0% do texto processado.

Para além disso, as métricas de precision, recall e F1-Score também são mais altas no texto lematizado (92,23%, 92,50% e 92,36%, respectivamente) em relação ao texto processado (88,1%, 88,0% e 88,0%). Apesar do *overfitting* ligeiro no treino (99,2% vs. 91,99% no teste), a lematização parece ter melhorado a generalização do modelo.

Capítulo 5

Desenvolvimento da Interface de Utilizador (UI)

O desenvolvimento da interface de utilizador (UI) teve como objetivo principal proporcionar uma interação intuitiva, acessível e eficiente com os modelos de classificação de sentimentos e emoções desenvolvidos no âmbito do projeto. A interface gráfica foi concebida para permitir que utilizadores, mesmo sem conhecimentos técnicos aprofundados, possam interagir diretamente com os modelos, inserindo textos e obtendo resultados de forma clara e imediata.

Para alcançar este objetivo, foi selecionada a biblioteca **Gradio**, reconhecida pela sua simplicidade, flexibilidade e capacidade de integração com modelos implementados em *Python*, facilitando a criação de uma *interface* interativa.

5.1 Tecnologias Utilizadas

A construção da *interface* baseou-se num conjunto de tecnologias amplamente utilizadas no desenvolvimento de aplicações de *machine learning* e *web interfaces*.

As principais tecnologias aplicadas foram:

- **Python 3.11:** Linguagem de programação principal do projeto, escolhida pela sua versatilidade, ampla comunidade e suporte robusto para bibliotecas de *machine learning* e desenvolvimento de interfaces.
- **Gradio:** Biblioteca *open-source* que permite a criação de interfaces web interativas simples. Esta é particularmente adequada para projetos de *machine learning*, pois simplifica a integração de modelos treinados com *interfaces* acessíveis pelo navegador, suportando entradas de texto, seleção de opções e visualização de resultados em tempo real.
- **Joblib:** Biblioteca utilizada para a serialização e carregamento dos modelos e vectorizadores. A escolha do *Joblib* deve-se à sua capacidade de lidar com objetos Python complexos, como modelos treinados, garantindo rapidez no carregamento e economia de recursos computacionais.

5.2 Arquitetura da Solução

A arquitetura da aplicação foi projetada com foco na modularidade, escalabilidade e usabilidade, garantindo que cada componente desempenhasse uma função específica e que a interação entre eles fosse fluida. A solução foi estruturada nas seguintes etapas:

1. **Pré-Processamento:** Definição de uma função de pré-processamento comum para limpar e normalizar o texto introduzido pelo utilizador. Esta função inclui as etapas mencionadas

na Secção 2.3.3. Esta etapa é essencial para garantir que os dados de entrada estejam no mesmo formato utilizado durante o treino dos modelos, assegurando consistência nos resultados.

2. **Carregamento dos Modelos:** Dois modelos distintos foram integrados na aplicação: um para classificação de sentimentos e outro para classificação de emoções. Cada modelo foi acompanhado pelo seu respectivo vectorizador, responsável por transformar o texto em representações numéricas (como vetores TF-IDF ou *embeddings*). O uso do `Joblib` garantiu que o carregamento dos modelos fosse eficiente, minimizando o tempo de inicialização da aplicação.
3. **Função de Previsão:** Uma função central foi implementada para integrar os modelos e realizar previsões. Esta função processa o texto pré-processado, seleciona o modelo apropriado com base na tarefa escolhida (sentimento ou emoção) e retorna a classe prevista juntamente com as probabilidades associadas. A inclusão das probabilidades oferece ao utilizador uma visão mais detalhada sobre a confiança do modelo na sua previsão, promovendo maior transparência.
4. **Interface com Gradio:** A interface foi construída utilizando o *framework Gradio*, que proporciona uma experiência de utilizador interativa e visualmente amigável. A interface permite:
 - Um campo de texto onde o utilizador pode inserir frases ou parágrafos que pretenda analisar.
 - Escolha entre análise de sentimentos ou emoções, garantindo flexibilidade na utilização dos modelos.
 - Exibição da classe prevista e da probabilidade associada, apresentada em formato numérico ou gráfico, dependendo da configuração escolhida.



Figura 5.1: Visualização da Página

A arquitetura foi desenhada para ser modular, permitindo a adição de novos modelos ou funcionalidades no futuro sem a necessidade de reformular a aplicação. Além disso, a separação clara entre pré-processamento, carregamento de modelos e interface garante facilidade de manutenção e escalabilidade.

5.3 Desafios e Considerações

O desenvolvimento da *interface* de utilizador enfrentou diversos desafios técnicos e conceituais que exigiram soluções cuidadosamente planeadas de maneira a garantir a funcionalidade, eficiência e usabilidade da aplicação.

Um dos principais obstáculos foi assegurar a consistência entre os dados de entrada fornecidos pelo utilizador e os modelos de *machine learning* utilizados. Para que as previsões fossem precisas, era essencial que o pré-processamento aplicado aos textos introduzidos replicasse exatamente as etapas realizadas durante o treinamento dos modelos.

Outro aspecto crítico foi a otimização do desempenho da aplicação, especialmente no que diz respeito ao tempo de resposta. O carregamento de modelos de *machine learning* e os seus respectivos vectorizadores pode ser uma tarefa computacionalmente intensiva, representando um risco para a experiência do utilizador, especialmente em dispositivos com recursos limitados.

Para mitigar esse problema, a biblioteca *Joblib* foi utilizada para realizar a serialização eficiente dos modelos, reduzindo o tempo necessário para carregá-los na memória. Além disso, a interface foi projetada com o objetivo de minimizar o consumo de recursos, mantendo a leveza e a responsividade, mesmo durante o processamento de textos complexos ou em cenários de uso intensivo.

A usabilidade da *interface* também foi uma prioridade central no desenvolvimento, considerando que o público-alvo incluía desde investigadores com conhecimento técnico até utilizadores sem qualquer experiência em *machine learning*. Para atender a essa diversidade de utilizadores, a *interface* foi projetada para ser intuitiva, oferecendo instruções claras e *feedback* visual imediato.

Por fim, a escalabilidade da aplicação foi um fator considerado desde o início do desenvolvimento. A sua arquitetura foi planeada para suportar a integração de novos modelos ou tarefas no futuro, como a inclusão de suporte a novos idiomas ou a análise de outros tipos de dados textuais. A escolha do *framework Gradio*, combinada com a modularidade da arquitetura, permite que novas funcionalidades sejam adicionadas com alterações mínimas no código existente, garantindo que a aplicação possa evoluir de acordo com as necessidades futuras do projeto.

5.4 Testes

A aplicação foi testada localmente num conjunto diversificado de frases, de maneira a garantir robustez, precisão e usabilidade. Os principais objetivos destes testes foram:

- Verificar se as classes previstas (sentimento ou emoção) estavam consistentes com o conteúdo do texto introduzido.
- Testar a alternância entre as tarefas de classificação de sentimentos e emoções, garantindo que o modelo e o vectorizador corretos fossem carregados dinamicamente sem erros.
- Avaliar se as probabilidades associadas às previsões eram exibidas corretamente e se refletiam a incerteza do modelo em casos ambíguos ou complexos.

Capítulo 6

Resultados e Discussão

6.1 Objetivos

Nas semanas 10 a 12, o foco do projeto esteve no refinamento do modelo, na sua avaliação final e na preparação da apresentação e documentação do trabalho realizado. O objetivo foi melhorar o desempenho do modelo com base na análise de erros, realizar testes finais para validar os resultados e organizar todo o projeto para a sua apresentação.

Na décima semana, foram analisados os erros cometidos pelo modelo e identificadas possíveis melhorias, como a adição de novos atributos, ajustes no pré-processamento e otimizações nos parâmetros do modelo. Além disso, foram exploradas formas de lidar com limitações, como viés nos dados ou especificidade do domínio, garantindo um modelo mais robusto e generalizável.

A décima primeira semana centrou-se na avaliação final do modelo, utilizando o conjunto de teste não visto anteriormente. Foram calculadas métricas finais de desempenho, como *accuracy*, precisão, *recall* e *F1-score*, comparando os resultados obtidos com a linha de base e com outros trabalhos publicados. Esta análise permitiu documentar as forças e limitações do modelo desenvolvido.

Por fim, na décima segunda semana, o foco esteve na preparação da apresentação e na documentação do projeto. Foi elaborada uma apresentação resumindo os objetivos, metodologias, resultados e desafios do trabalho. Além disso, foi produzido um relatório final detalhado, incluindo todas as etapas do projeto, desde a aquisição dos dados até a avaliação dos modelos. A prática da apresentação e a recolha de *feedback* ajudaram a garantir uma comunicação clara e eficaz dos resultados obtidos.

Este capítulo apresenta o refinamento do modelo, a sua avaliação final e o processo de preparação da apresentação e documentação do projeto, consolidando todo o trabalho realizado ao longo das semanas anteriores.

6.2 Comparaçao entre Datasets

Discute diferenças de desempenho e atributos entre os *datasets*.

Com o objetivo de tornar o sistema de classificação acessível e funcional para utilizadores não técnicos, foi desenvolvida uma interface gráfica interativa utilizando a biblioteca **Gradio**. Esta ferramenta permite criar rapidamente aplicações web simples e eficazes para interação com modelos de *machine learning*.

A aplicação final oferece ao utilizador três funcionalidades principais:

- **Previsão da Emoção:** identifica a emoção principal associada à frase introduzida (por exemplo: alegria, tristeza, raiva, etc.).
- **Previsão do Sentimento:** classifica a frase como positiva ou negativa.
- **Guardar o resultado obtido:** guarda a frase, a previsão escolhida associada e a data da operação numa pasta própria.

← → ⌛ 127.0.0.1:7860

Previsão de Emoções e Sentimentos em frases

Escreva uma frase e escolha se quer prever a emoção ou o sentimento associado.

text
I am happy!

Escolha o tipo de previsão

 Emoção Sentimento

Emoção prevista: Joy

- Confiança da previsão: 47,58%

Flag as "Guardar resultado"

Clear
Submit

[Usar via API](#) 🔍 · [Construído com Gradio](#) 🚀 · [Configurações](#) 🎞

Previsão da Emoção

← → ⌛ 127.0.0.1:7860

Previsão de Emoções e Sentimentos em frases

Escreva uma frase e escolha se quer prever a emoção ou o sentimento associado.

text
I am happy!

Escolha o tipo de previsão

 Emoção Sentimento

Sentimento prevista: Positive

- Confiança da previsão: 99,41%

Flag as "Guardar resultado"

Clear
Submit

[Usar via API](#) 🔍 · [Construído com Gradio](#) 🚀 · [Configurações](#) 🎞

Previsão do Sentimento

Além disso, a interface exibe a **probabilidade (confiança)** da previsão feita, permitindo ao utilizador perceber o grau de certeza do modelo para a classificação apresentada.

Funcionamento

1. **Input do Utilizador:** o utilizador insere uma frase num campo de texto. Este texto é submetido a um processo de pré-processamento (remoção de pontuação, links, normalização, etc.), garantindo que a entrada esteja em conformidade com os dados usados no treino dos modelos.
2. **Escolha do Tipo de Análise:** o utilizador pode escolher entre:
 - **Classificação de Emoções**
 - **Classificação de Sentimentos**A aplicação carrega automaticamente o modelo e o vectorizador correspondente a cada tarefa, previamente guardados com `joblib`.
3. **Apresentação do Resultado:** após a previsão, é apresentada:
 - A classe prevista
 - A precisão associada à previsão (probabilidade da classe prevista, obtida via `predict_proba()`)

Estrutura Técnica

A implementação segue o seguinte pipeline:

- Modelos previamente treinados e guardados em ficheiros `.pkl`
- Vectorizadores TF-IDF ou BoW, consoante o melhor desempenho observado na fase de experimentação
- Funções de pré-processamento aplicadas a toda a entrada textual
- Interface criada com `gr.Interface`, permitindo a execução local ou via navegador

Utilidade

Esta interface serve como uma demonstração interativa do sistema desenvolvido, e num contexto real poderia também servir como uma ferramenta útil, como por exemplo em:

- Análise de *feedback* de clientes
- Monitorização de redes sociais
- Estudos de comportamento emocional em linguagem natural

A simplicidade da interface, aliada à robustez dos modelos, oferece uma forma eficaz de aplicar os resultados do projeto em cenários reais, com uma usabilidade adequada mesmo para utilizadores não especializados.

Capítulo 7

Conclusão

Ao longo das doze semanas de desenvolvimento deste projeto, foi possível evoluir desde a fase inicial de exploração e aquisição de dados até à experimentação avançada com diferentes técnicas de representação textual e algoritmos de *machine learning* aplicados à análise de sentimentos.

Uma das combinações de técnicas mais relevantes analisadas foi **BoW + TF-IDF + GloVe**, que conjugou a eficácia das abordagens baseadas em frequência com a capacidade de representação semântica dos *embeddings* pré-treinados do *GloVe*. Esta combinação revelou-se particularmente eficaz quando associada ao classificador *Support Vector Machines* (SVM), atingindo uma **accuracy de 0.906**, colocando-se entre os melhores desempenhos observados no estudo.

A integração de *embeddings* semânticos permitiu ao modelo captar relações entre palavras que a simples contagem de termos não consegue representar, beneficiando a capacidade do classificador em lidar com sinônimos e variações lexicais. Contudo, apesar destes benefícios, a performance do modelo foi ligeiramente inferior à de outras combinações que incluíam *FastText*, sugerindo que a escolha do *embedding* influencia diretamente a eficácia da representação textual.

Durante o processo de engenharia de atributos e otimização, verificou-se também que modelos baseados unicamente em *embeddings* (como *Word2Vec* ou *GloVe* isolados) apresentaram desempenhos inferiores, reforçando a importância da fusão entre informação estatística e semântica para uma classificação mais robusta.

A metodologia seguida, sustentada por um pipeline modular e sistemático — que incluiu recolha, limpeza, representação, modelação e avaliação — possibilitou uma análise comparativa abrangente, apoiada por métricas quantitativas (como *accuracy*, *precision* e *F1-score*) e visualizações informativas. Estes resultados permitiram tirar conclusões sólidas sobre o impacto das diferentes combinações de técnicas na performance dos modelos, contribuindo para uma compreensão mais profunda das boas práticas em classificação de sentimentos.

Bibliografia

- [1] Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed., Draft). Stanford University. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>
- [2] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. Disponível em: <https://nlp.stanford.edu/IR-book/>
- [3] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Disponível em: <https://aclanthology.org/D14-1162/>
- [4] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint arXiv:1301.3781. Disponível em: <https://arxiv.org/abs/1301.3781>
- [5] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. Disponível em: <https://jmlr.org/papers/v12/pedregosa11a.html>
- [6] Řehůřek, R., & Sojka, P. (2010). *Software Framework for Topic Modelling with Large Corpora*. LREC 2010 Workshop on New Challenges for NLP Frameworks. Disponível em: <https://radimrehurek.com/gensim/>