Instituto Federal de Goiás

Pós Graduação em Inteligência Artificial Aplicada

Disciplina: Processamento de Linguagem Natural

Professor: Daniel Xavier de Souza

Grupo: 4

Alunos: Wagner Silva, Cleibson, Marcos Rodrigues Brugnaro

Projeto Prático - Disciplina PLN/2023

Relatório Detalhado de Resultados

Pré-processamento das bases: Buscapé, B2W

1 - Dataset: Base buscapé:

1.1 - Acesso ao dataset:

Link:

https://www.kaggle.com/datasets/fredericods/ptbr-sentiment-analysis-datasets?select=buscape.csv

1.2 - Verificação de quantidades de instâncias e classes (features), do dataset original:

- Instâncias: 84.991;
- Classes (features): 5;

1.3 - Definição da Tarefa: Classificação de ratings das avaliações de clientes:

- 4 e 5: Positiva;
- 1 e 2: Negativa;
- 3: Descartadas;
- Tendo em vista estas regras, avaliações positivas recebem valor 1 e negativas recebem valor 0

1.4 - Remoção de instâncias nulas:

Após a importação do dataset, identificamos uma instância nula na feature review_text, removemos a mesma:

```
review_text     True
rating         False
dtype: bool
```

	review_text	rating
38852	NaN	4

1.5 - Padronização do texto em lowercase(letras minúsculas):

```
def to_lowercase(text):
    text = text.apply(lambda x: " ".join(x.lower().split()) if isinstance(x, str) else "")
    return text
```

1.6 - Remoção de caracteres especiais e substituição de cedilhas por c:

```
def preprocess_text(text):
    # removendo caracteres especiais
    text = re.sub(r'[^\w\s]', '', text)
    # unidecode translitera caracteres ace
    text = unidecode(text)
    return text
```

1.7 - Remoção de hiperlinks:

```
def remove_hiperlink(text):
    text = re.sub(r"(https?://[^\s]+|www\.[^\s]+)", '', text)
    return text
```

1.8 - Descartando instâncias com avaliação igual a 3 :

```
rating_3 = df['rating'] == 3
rating_3.sum()
```

```
11364
```

```
df = df[df['rating'] != 3]
```

```
df.shape
```

```
(73626, 2)
```

1.9 - Ajustando ratings:

- 1 e 2 = 0
- 4 e 5 = 1

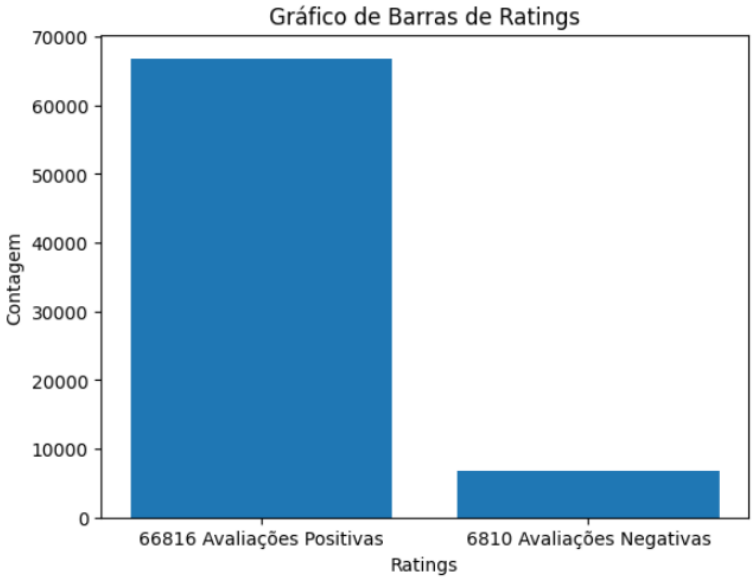
```
df['rating'] = list(map(lambda x: 0 if x<=2 else 1, df['rating']))
```

1.10 - Balanceamento das classes:

Identificamos que as classes do dataset buscapé estão desbalanceadas, 66.816 avaliações positivas, 6.810 avaliações negativas;

Isto pode enviesar o treinamento do modelo, fazendo com que o mesmo não consiga aprender corretamente os padrões:

```
rating
1    66816
0     6810
Name: count, dtype: int64
```

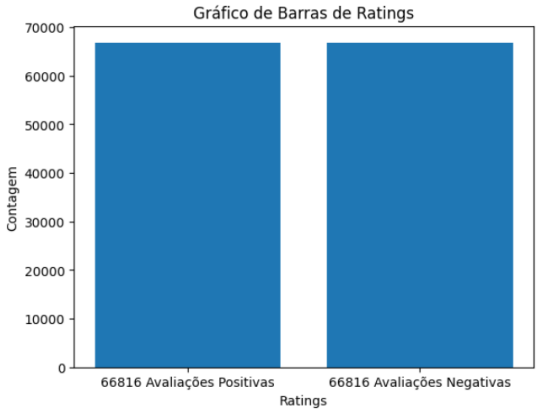


Para resolver este problema decidimos utilizar a técnica de reamostragem de dados chamada RandomOverSampler.

- Esta técnica gera novas amostras aleatórias para classe minoritária;
- Desta forma ela tenta igualar ao número de amostras da classe majoritária;

Resultados após o balanceamento das classes:

```
rating
1    66816
0    66816
Name: count, dtype: int64
```



- 66.816 avaliações positivas e negativas;

1.11 - Salvando resultados em um dataframe, que iremos utilizar durante os treinamentos.

```
preprocessed_balanced_df.to_csv('../datasets/buscape_preprocessed.csv', index=False)
```

Após o pré-processamento e balanceamento, a base buscapé aumentou a quantidade de instâncias para 133.632.

---

2 - Dataset: B2W:

2.1 - Acesso ao dataset:

Link: https://www.kaggle.com/datasets/fredericods/ptbr-sentiment-analysis-datasets?select=b2w.csv

2.2 - Verificação de quantidades de instâncias e classes (features), do dataset original:

- Instâncias: 132.373;
- Classes (features): 5;

2.3 - Definição da Tarefa: Classificação de ratings das avaliações de clientes:

- 4 e 5: Positiva;
- 1 e 2: Negativa;
- 3: Descartadas;
- Tendo em vista estas regras, avaliações positivas recebem valor 1 e negativas recebem valor 0

2.4 - Remoção de instâncias nulas:

- Não ocorreram instâncias nulas;

```
review_text    False
rating         False
dtype: bool
```

2.5 - Padronização do texto em lowercase(letras minúsculas):

```
def to_lowercase(text):
    text = text.apply(lambda x: " ".join(x.lower().split()) if isinstance(x, str) else "")
    return text
```

2.6 - Remoção de caracteres especiais e substituição de cedilhas por c:

```
def preprocess_text(text):
    # removendo caracteres especiais
    text = re.sub(r'[^\w\s]', '', text)
    # unidecode translitera caracteres ace
    text = unidecode(text)
    return text
```

2.7 - Remoção de hiperlinks:

```
def remove_hiperlink(text):
    text = re.sub(r"(https?://[^\s]+|www\.[^\s]+)", '', text)
    return text
```

2.8 - Descartando instâncias com avaliação igual a 3 :

```
rating_3 = df['rating'] == 3
rating_3.sum()
```

```
11364
```

```
df = df[df['rating'] != 3]
```

```
df.shape
```

```
(73626, 2)
```

2.9 - Ajustando ratings:

- 1 e 2 = 0
- 4 e 5 = 1

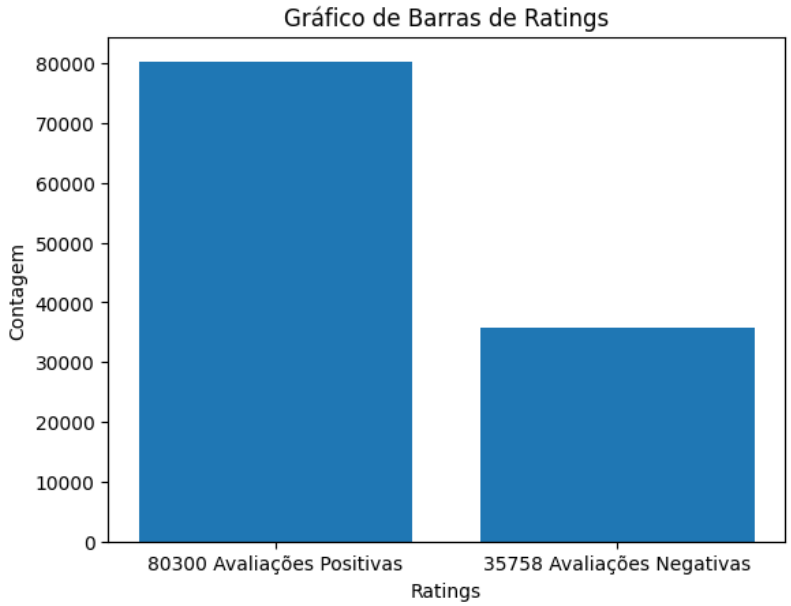
```
df['rating'] = list(map(lambda x: 0 if x<=2 else 1, df['rating']))
```

2.10 - Balanceamento das classes:

Identificamos que as classes do dataset b2w estão desbalanceadas, 80.300 avaliações positivas, 35.758 avaliações negativas;

Isto pode enviesar o treinamento do modelo, fazendo com que o mesmo não consiga aprender corretamente os padrões:

```
rating
1    80300
0    35758
Name: count, dtype: int64
```

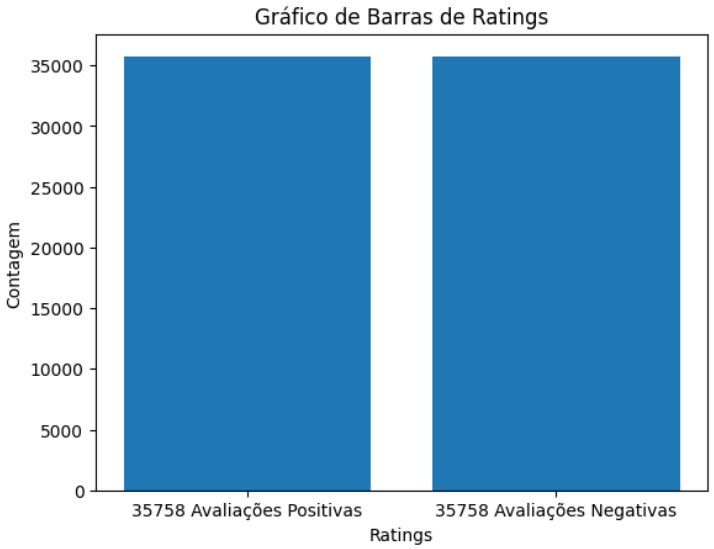


Para resolver este problema decidimos utilizar a técnica de reamostragem de dados chamada RandomUnderSampler.

- Desta forma ela tenta igualar ao número de amostras da classe majoritária com a classe minoritária;

Resultados após o balanceamento das classes:

```
rating
0    35758
1    35758
Name: count, dtype: int64
```



- 66.816 avaliações positivas e negativas;

2.11 - Salvando resultados em um dataframe, que iremos utilizar durante os treinamentos.

```
preprocessed_balanced_df.to_csv('../datasets/b2w_preprocessed.csv', index=False)
```

Após o pré-processamento e balanceamento, a base b2w diminuiu a quantidade de instâncias para 71.516.

---

3. Modelos Transformers:

Identificador: Ro_B

Modelo: XML-Roberta base

Multilingue: sim

Tamanho do Embedding: 768

Identificador: Ro_L

Modelo: XML-Roberta large

Multilingue: sim

Tamanho do Embedding: 1024

Identificador: Bert_B

Modelo: BerTimbau base

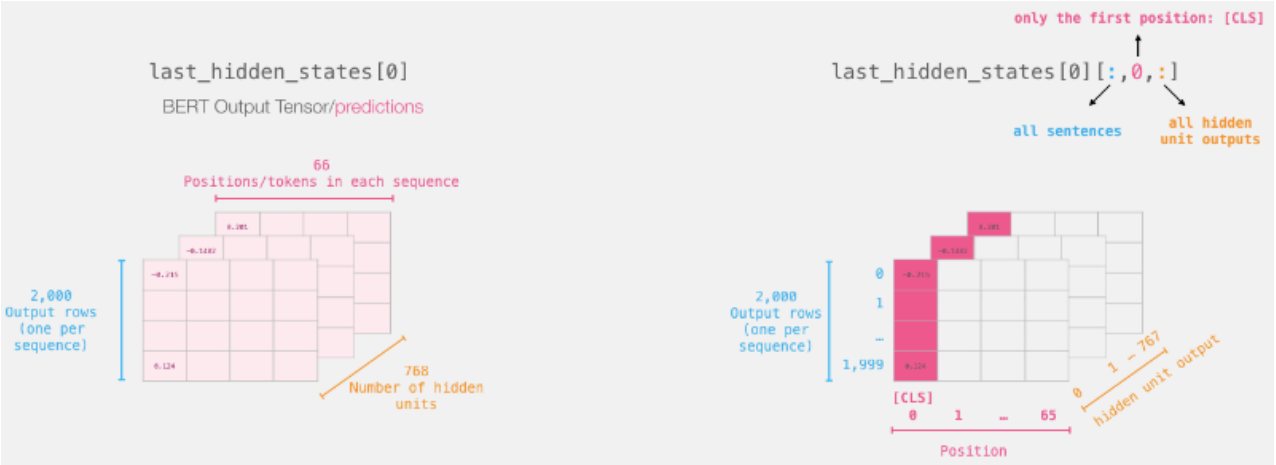
Multilingue: sim

Tamanho do Embedding: 768

---

4. Uso dos Embeddings como Feature Based:

- FB1: Concatena Média, Min, e Max;
- FB2: Concatena CLS + Média de todos + std;



---

5. Organização e detalhamento da execução dos Experimentos dos Embeddings Estáticos:

- **Definição dos modelos:**
- TFIDF + MLP
- GloVe + MLP
- GloVe + Fine Tuning
- FastText + MLP
- FastText + FineTuning
- **Variação dos seguintes hiperparâmetros:**
- Variação de Dropout: 5%, 15%, 25%, 35%, 50%
- Variação de Learning Rate: 5e-4, 1e-3, 5e-3, 1e-2
- **Hiperparâmetros que mantemos durante os treinamentos:**
- Cross-validation, 10 k-folds;
- Épocas: 5;
- **Arquitetura rede MLP:**
- Camada densa 24 neurônios, função relu;
- Dropout com sua respectiva variação;
- Camada densa com 1 neurônio, função de ativação sigmoid;
- Sumarizamos a rede para identificarmos total de parâmetros que vão ser treinados;

- **Exemplo de sequência das variações de hiperparâmetros para cada modelo:**
- Dropout 5%, Learning Rate: 5e-4
- Dropout 5%, Learning Rate: 1e-3
- Dropout 5%, Learning Rate: 5e-3
- Dropout 5%, Learning Rate: 1e-2
- **Iniciando outra sequência de variações:**
- Dropout 15%, Learning Rate: 5e-4
- **E segue a mesma lógica até finalizar as variações.**
- 5 sequências de variações, cada variação com 4 treinamentos, para cada um dos 5 modelos;
- Totalizando 100 treinamentos experimentais;

5.1 - Testes Estatísticos:

- Após a finalização do treinamento de cada sequência de variação em seu respectivo modelo e base, foram armazenados os valores das médias F1 Score e Acurácia;
- Documentamos os valores de cada um dos treinamentos nas tabelas:
- Testes Estatísticos - base buscapé e Testes Estatísticos - base b2w, segue em anexo a este documento;
- Posteriormente analisamos as métricas cada sequência de treinamentos documentados nestas tabelas citadas anteriormente;
- Escolhemos o melhor resultado de treinamento de cada modelo com sua respectiva variação e extraímos gráficos para auxiliar nas comparações com os melhores resultados de outros modelos

---

6 - Detalhamento das Bases de Dados:

6.1 - Consolidado de informações originais e após pré-processamento, binarização e balanceamento das bases:

Detalhamento das Bases de Dados			
Dataset Original			
Dataset	**Quantidade de Instâncias (Dataset Original)**	**Classes**	**Quantidade de Instâncias Nulas**
Buscapé	84.991	5	1
B2w	132.373	5	0

Detalhamento das Bases de Dados					
Dataset Pré Processado					
Dataset	**Binarização (Quantidade de Classes Positivas)**	**Binarização (Quantidade de Classes Negativas)**	**Técnica de balanceamento**	**Quantidade de Instâncias após Binariazação, Pré-processamento e Balanceamento**	**Quantidade de Classes após Binariazação, Pré-processamento e Balanceamento**
Buscapé	66.816	6.810	RandomOverSampler	133.632	2
B2w	80.300	35.758	RandomUnderSampler	71.516	2

---

7 - Análises Estatísticas das bases - Melhores Resultados - Base Buscapé:

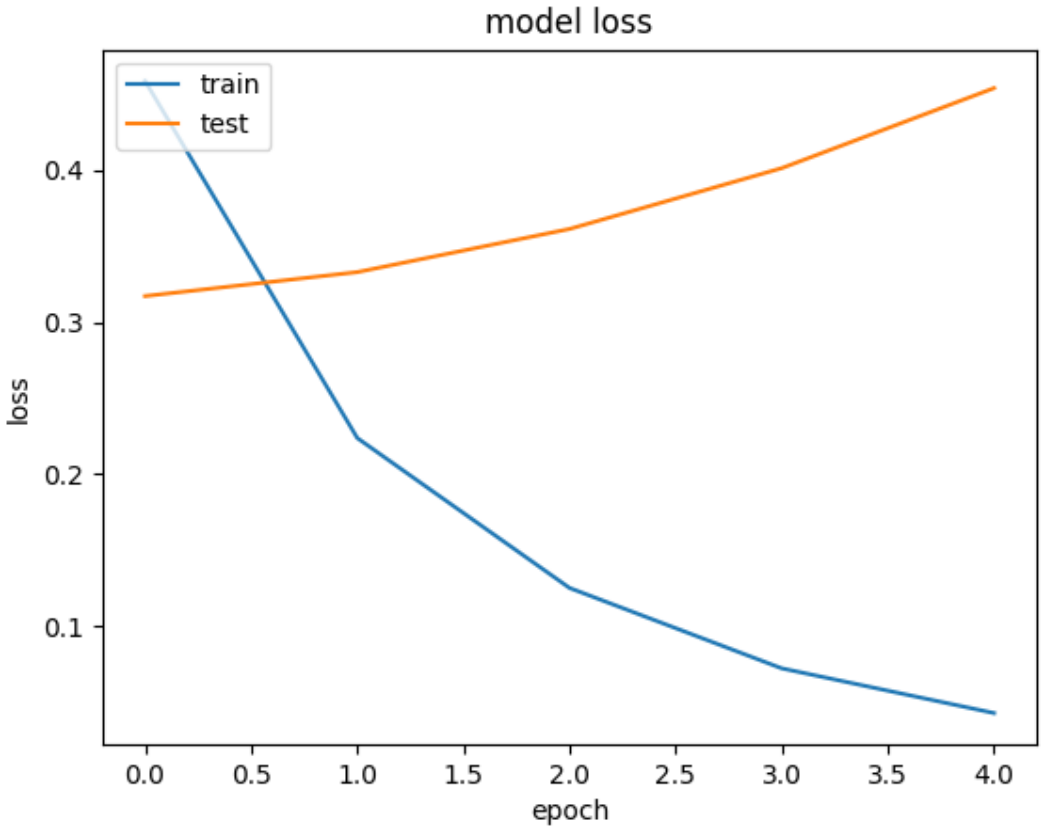
- Após os treinamentos de cada base com seus respectivos modelos, desenvolvemos um algoritmo que analisa as médias dos valores das métricas acurácia e f1 score dos treinamentos e retorna o melhor resultado de desempenho, e desvio padrão deste desempenho nas diferentes sequências de execuções;
- A seguir, análise estatística dos melhores resultados dos experimentos.

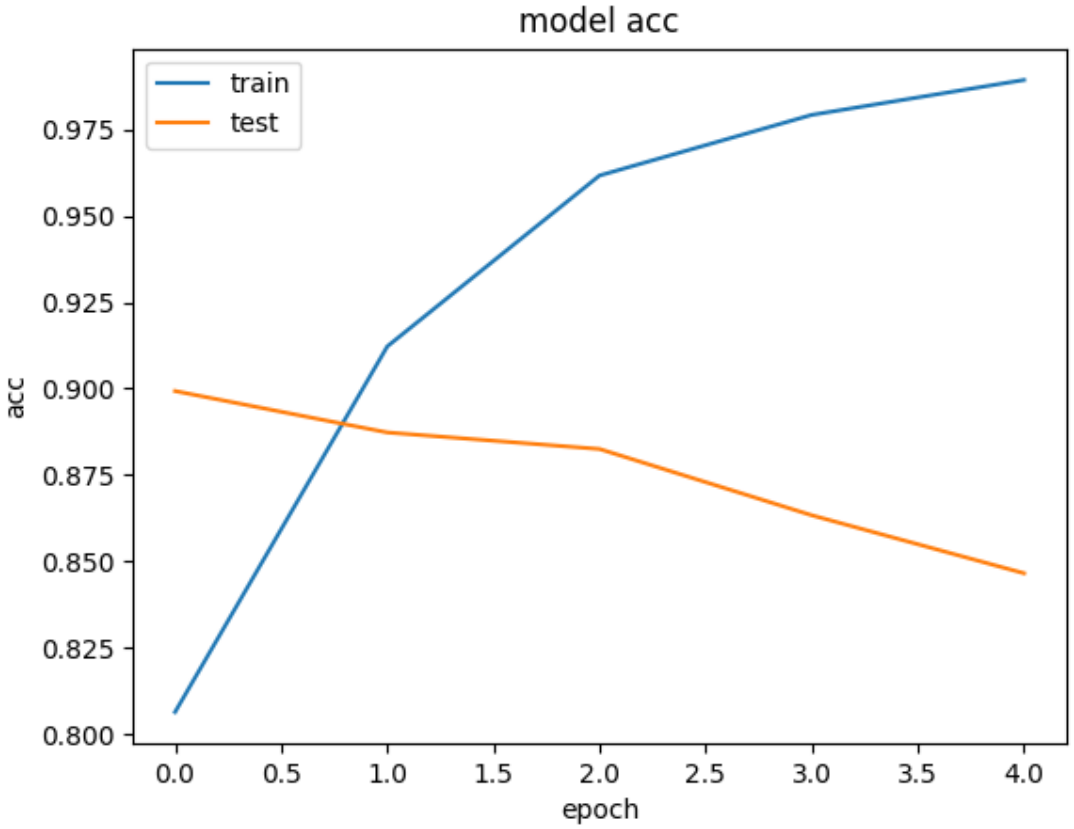
7.1 - Embeddings Estáticos, Modelo TFIDF + MLP:

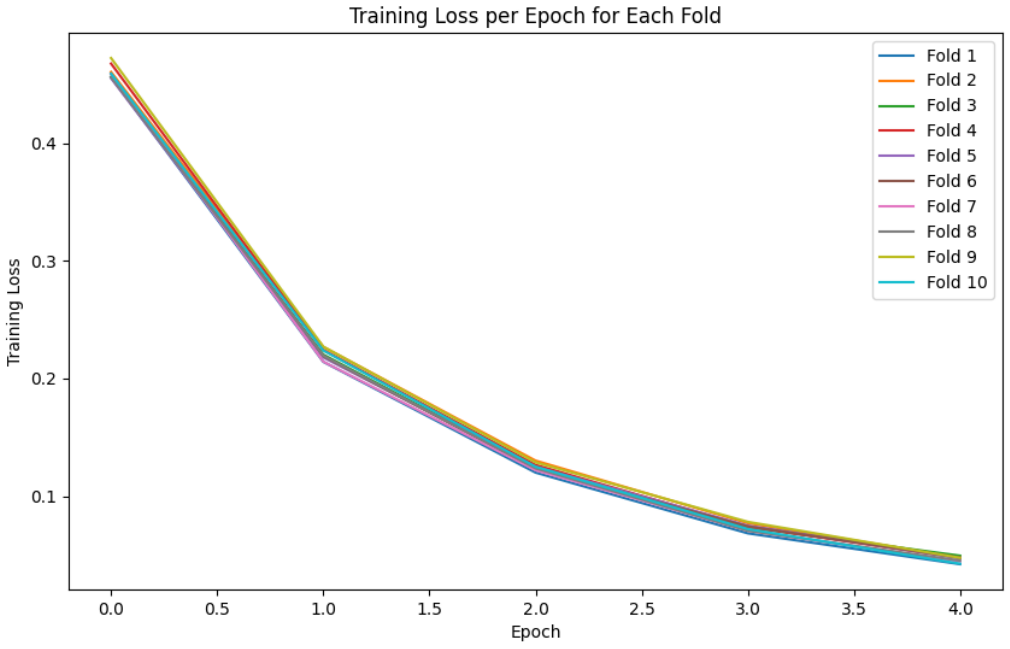
- Foram realizados experimentos com 12.5% das instâncias da base Buscapé, devido à sobrecarga de memória em treinamentos com quantidade acima de 12.5%;
- 12.5% equivale a 16.704 instâncias de um total de 133.632;
- Foram alterados os hiperparâmetros min_df e max_df da biblioteca TFIDFVectorizer para melhorar a performance do treinamento do modelo;

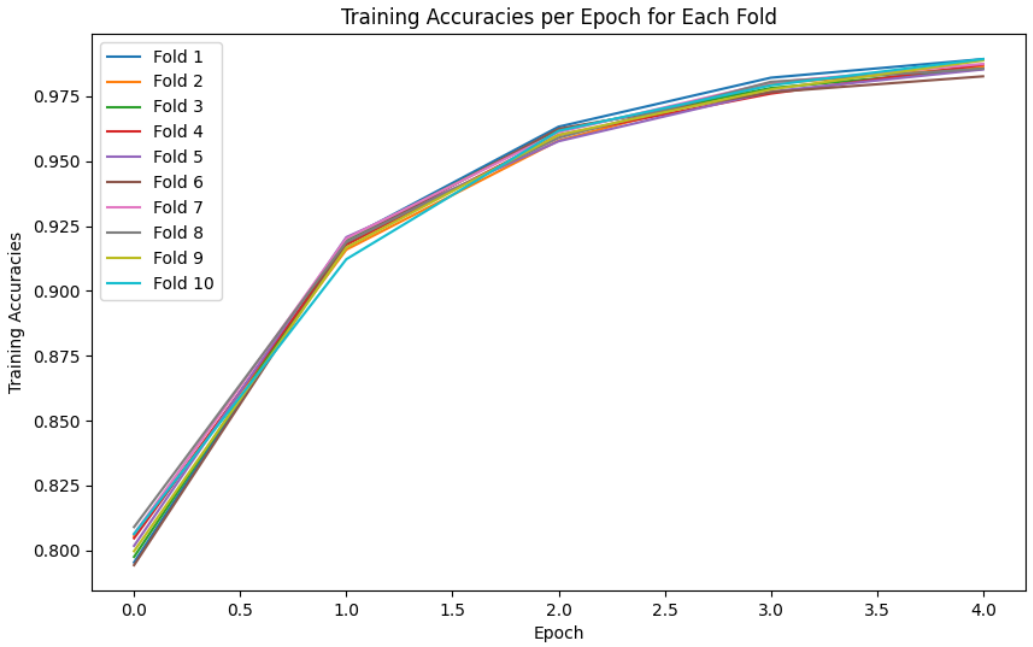
- Hiperparâmetro min_df, inicialmente começamos com valor 5 e durante alguns testes de performance aumentamos para 8. Este parâmetro ignora palavras com frequência menor que a definida no mesmo, em nosso caso 8;
- Hiperparâmetro max_df=0.3 foi adicionado, visando também a melhor performance do treinamento do modelo. Este parâmetro ignora os termos que tenham uma frequência de documento estritamente superior ao limite determinado.

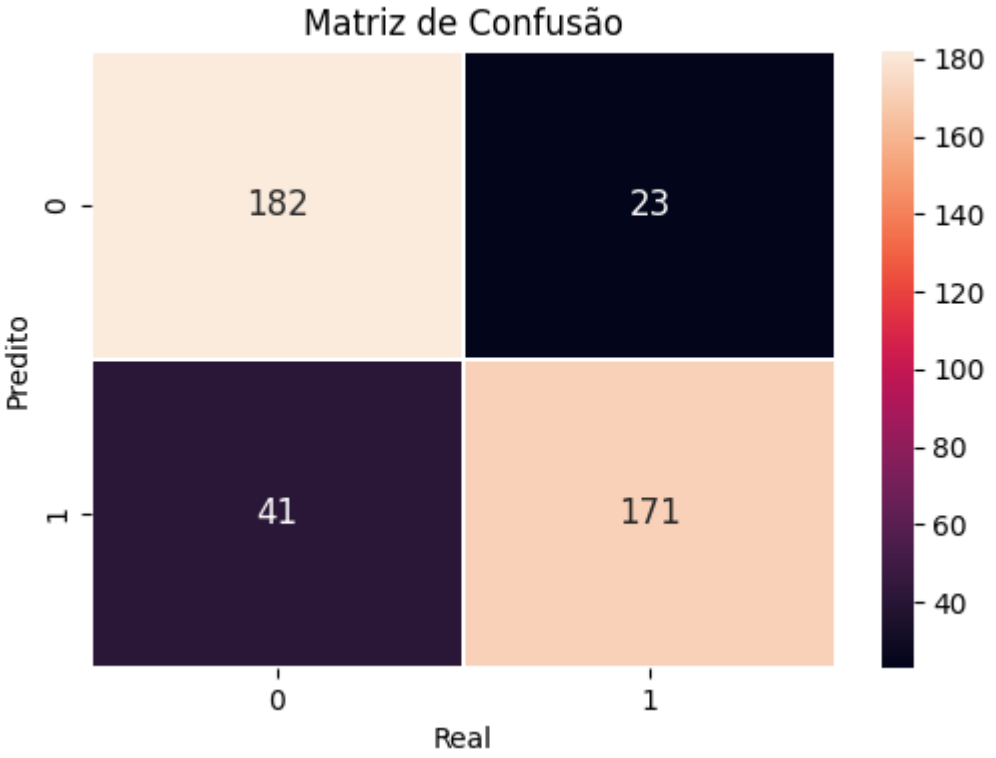
Modelo: TFIDF + MLP				
	Treino		Teste	
Hiperparâmetros	**F1 Score**	**Acurácia**	**F1 Score**	**Acurácia**
Dropout = 0.05, Learning rate = 5e-3	0.99 ± 0.02	0.95 ± 0.02	0.83 ± 0.01	0.83 ± 0.01




model acc
train
test
acc
epoch
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

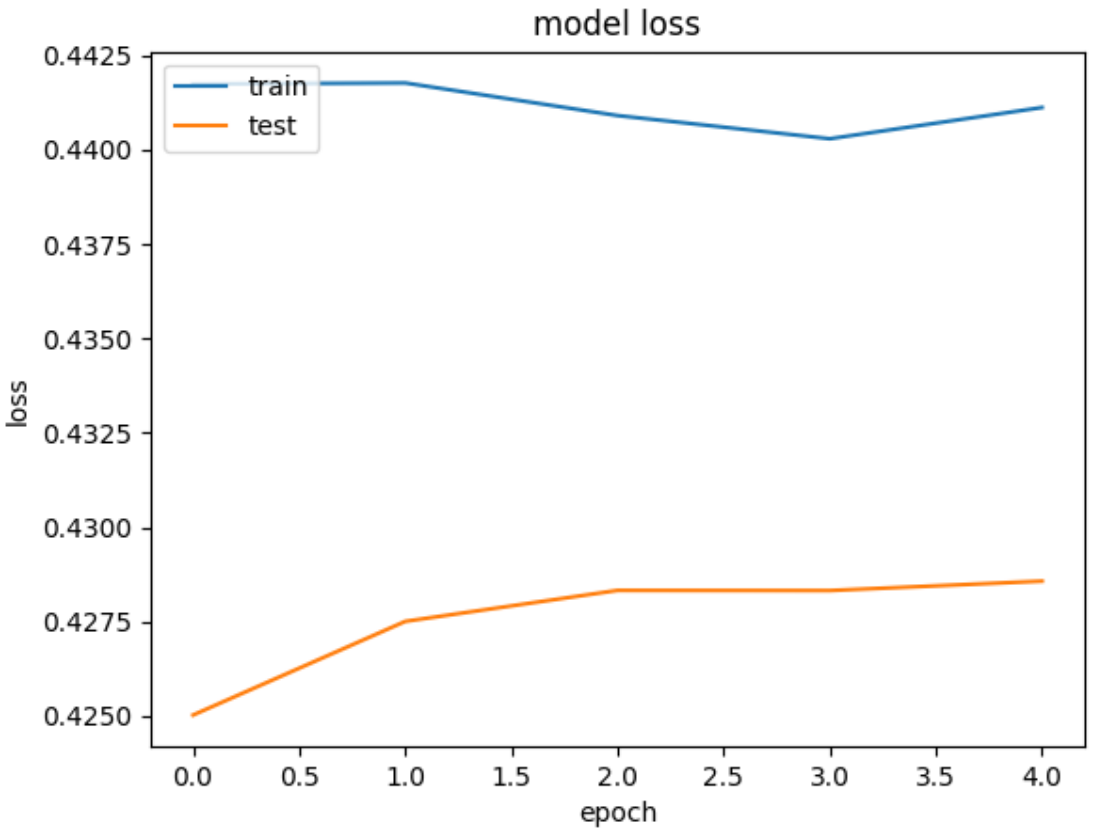

Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
Epoch
0.4
0.3
0.2
0.1
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

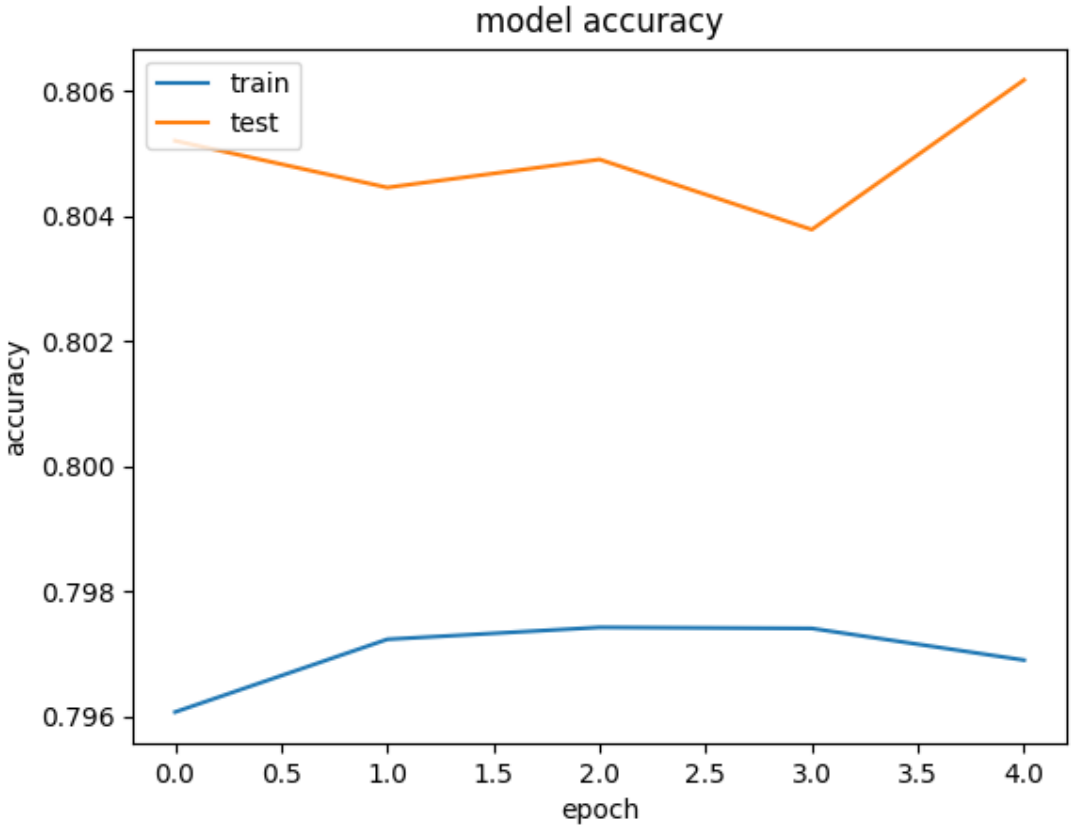

Training Accuracies per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Accuracies
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch

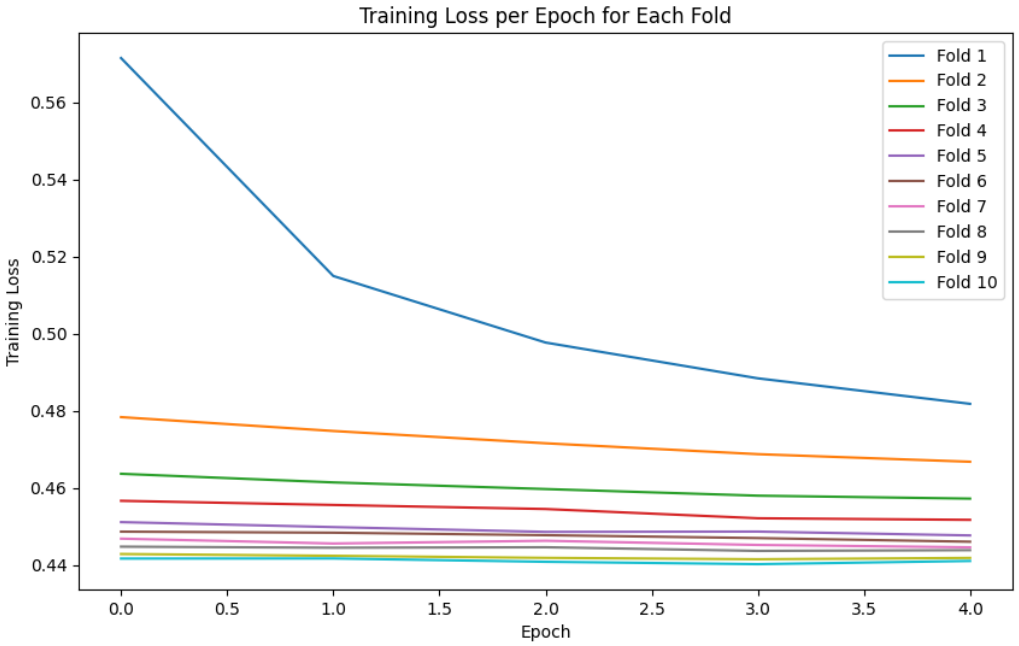

Matriz de Confusão
182
23
41
171
Predito
0
1
0
1
Real
180
160
140
120
100
80
60
40

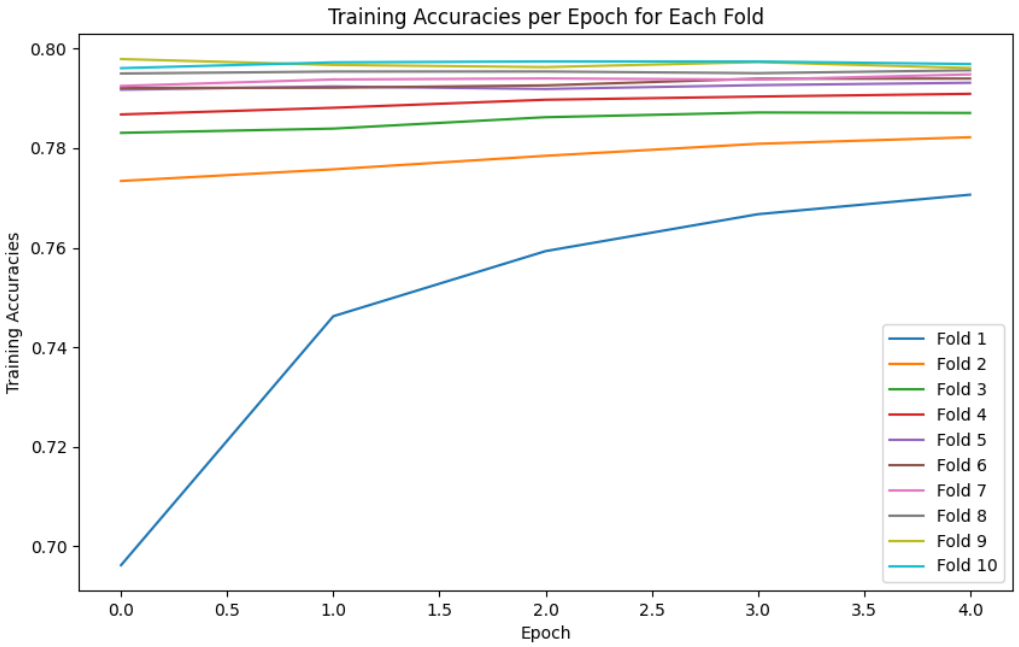
7.2 - Embeddings Estáticos, Modelo GloVe + MLP:

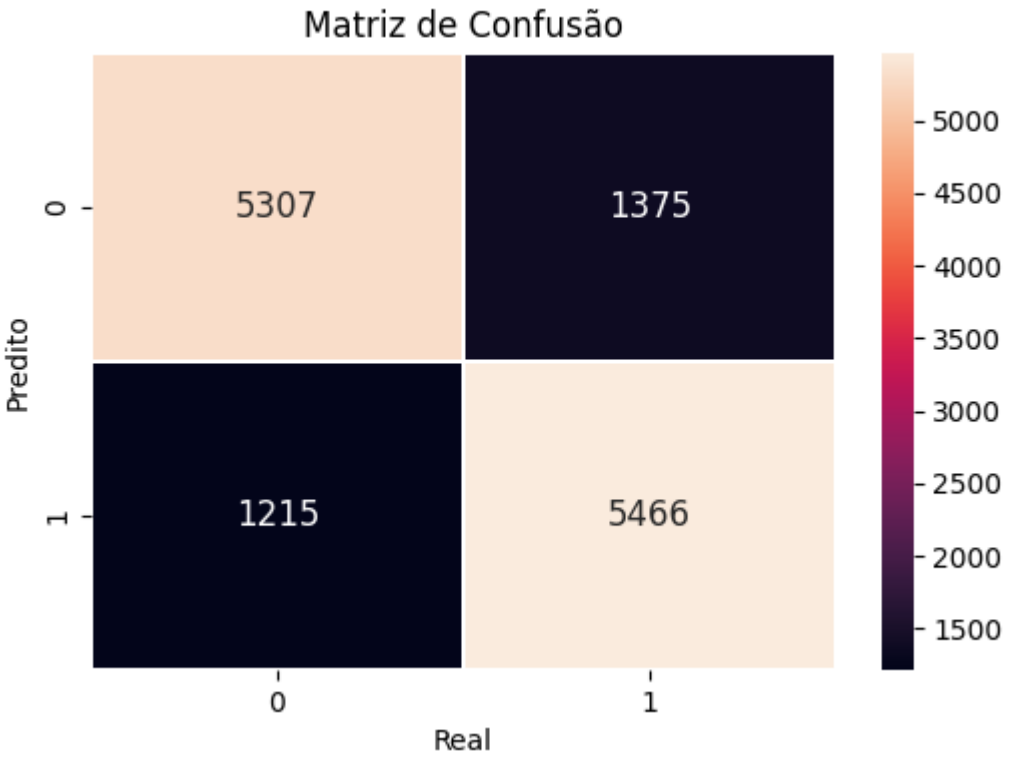
Modelo: GloVe + MLP				
	Treino		Teste	
Hiperparâmetros	**F1 Score**	**Acurácia**	**F1 Score**	**Acurácia**
Dropout = 0.05, Learning rate = 5e-4	0.80 ± 0.01	0.78 ± 0.01	0.79 ± 0.01	0.79 ± 0.01




model accuracy
train
test
accuracy
epoch
0.806
0.804
0.802
0.800
0.798
0.796
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

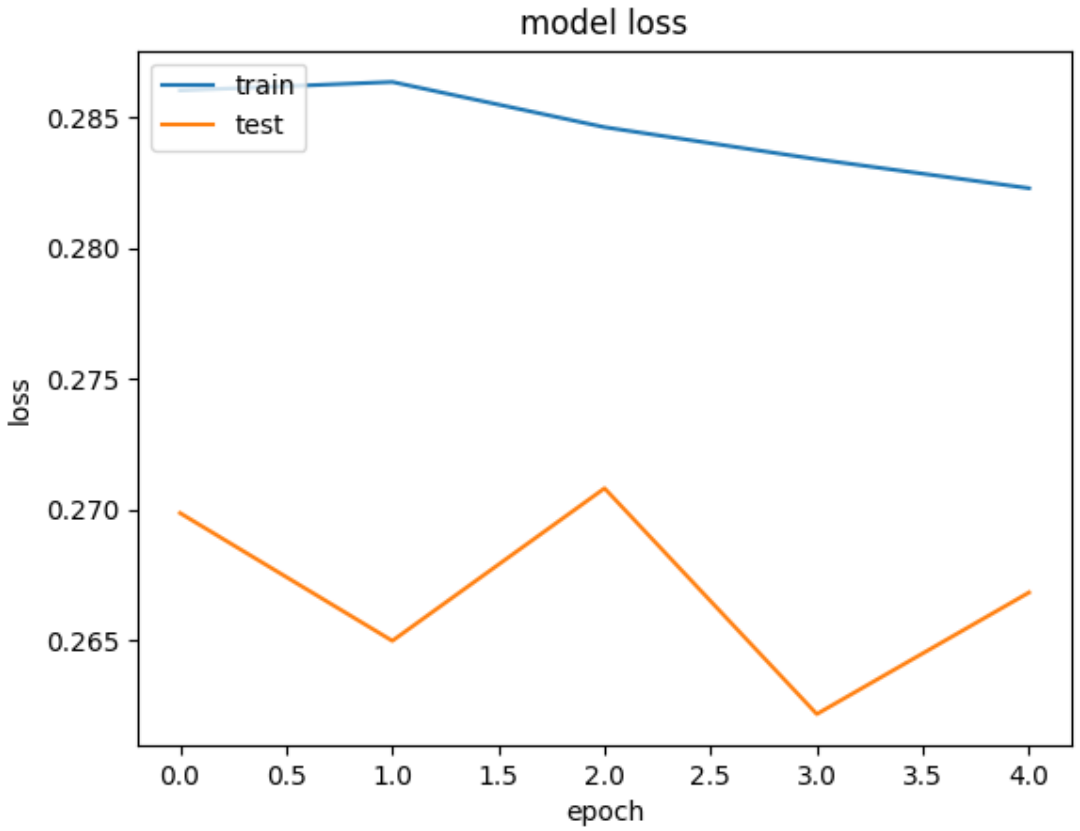

Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
Epoch
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

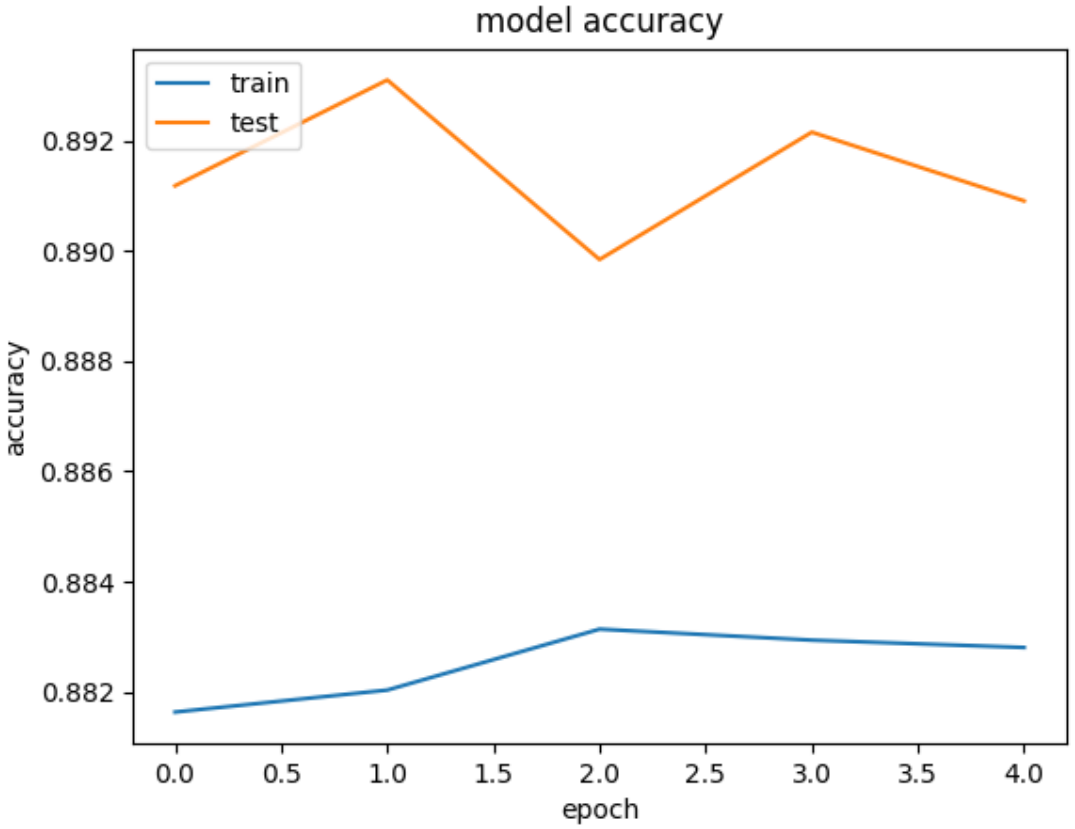

Training Accuracies per Epoch for Each Fold
Training Accuracies
Epoch
0.80
0.78
0.76
0.74
0.72
0.70
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

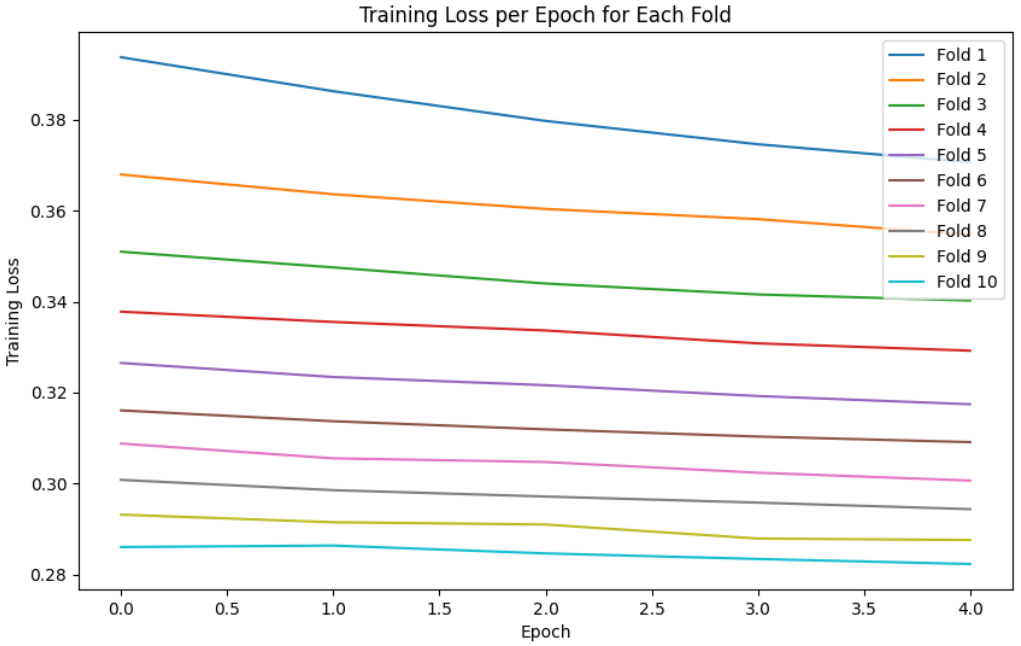

Matriz de Confusão
Predito
Real
5307
1375
1215
5466
0
1
0
1
5000
4500
4000
3500
3000
2500
2000
1500

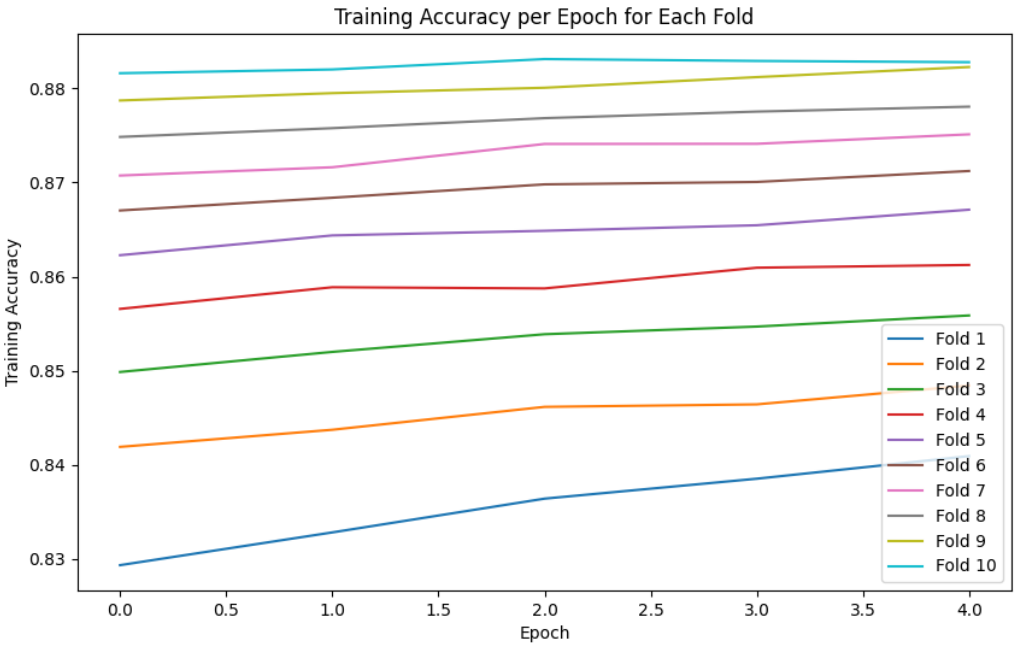
7.3 - Embeddings Estáticos, Modelo FastText + MLP:

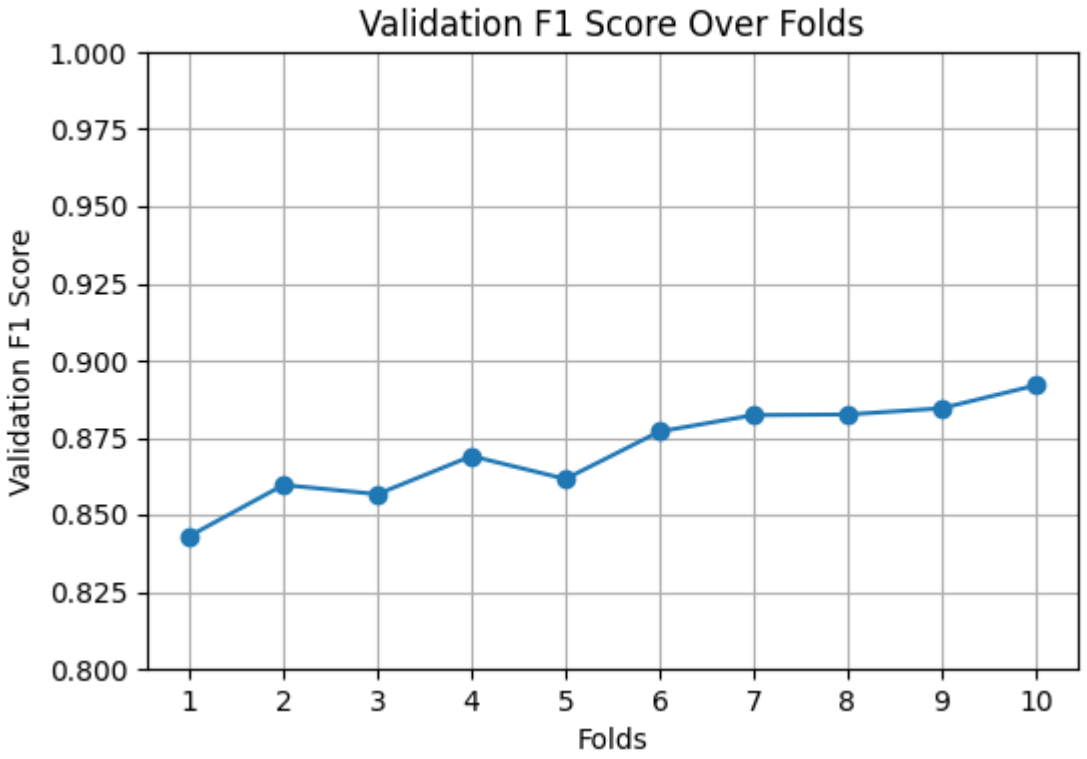
Modelo: FastText + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 5e-4	0.87 ± 0.01	0.86 ± 0.01	0.87 ± 0.01	0.86 ± 0.01

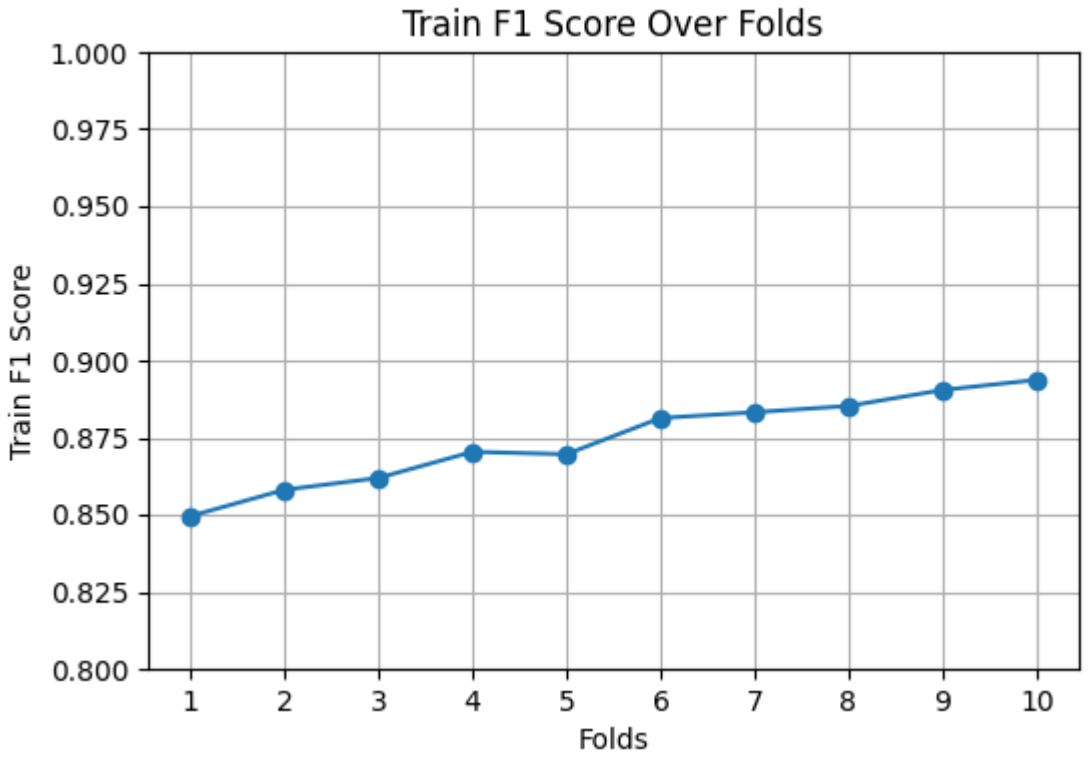


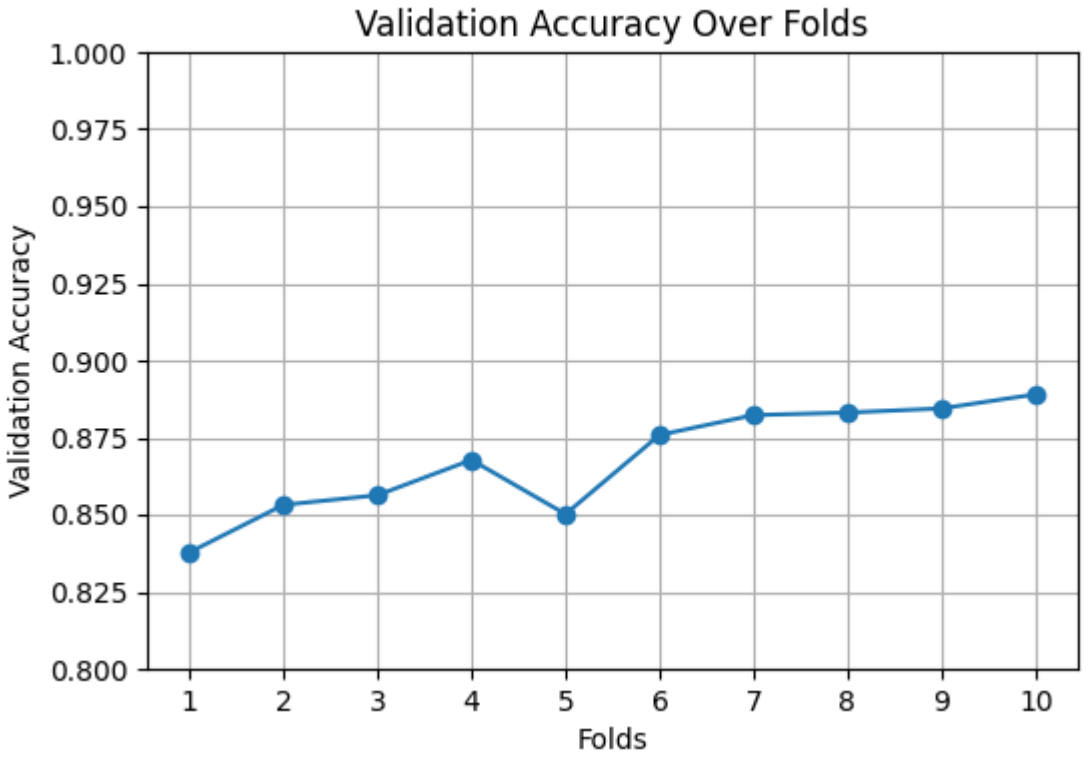

model accuracy
train
test
accuracy
0.892
0.890
0.888
0.886
0.884
0.882
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
0.38
0.36
0.34
0.32
0.30
0.28
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch


Training Accuracy per Epoch for Each Fold
Training Accuracy
Epoch
0.83
0.84
0.85
0.86
0.87
0.88
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

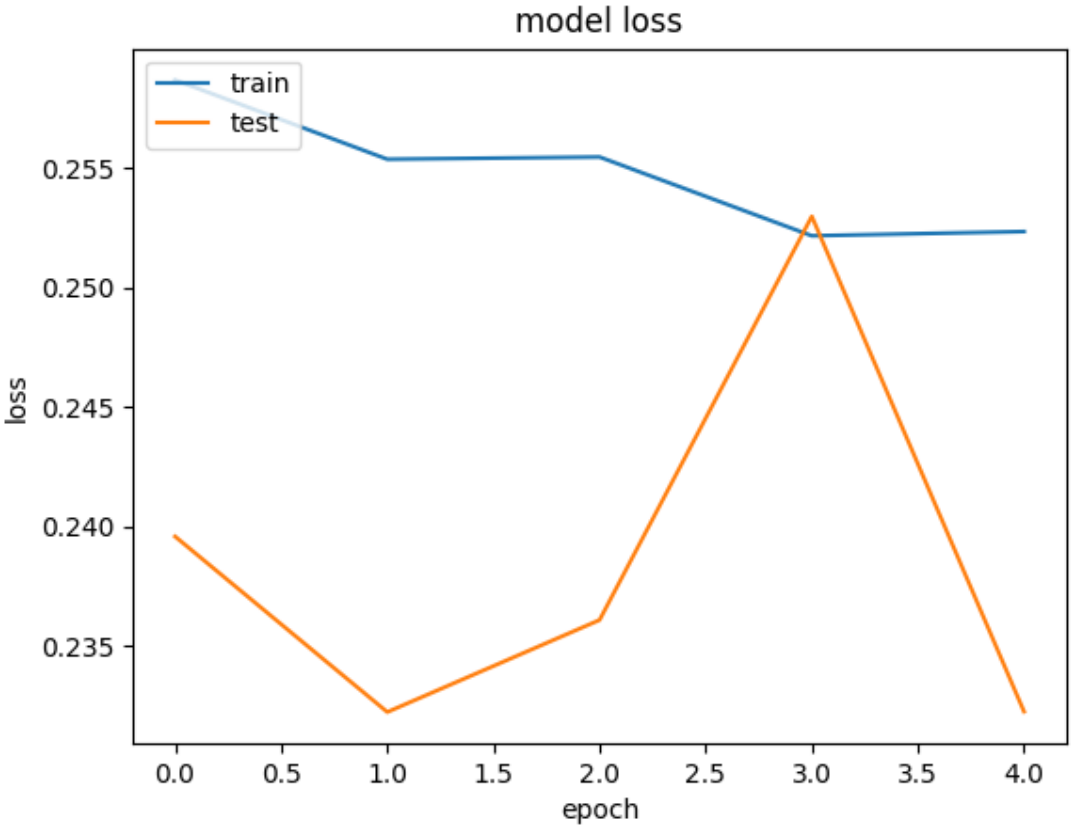

Validation F1 Score Over Folds
Validation F1 Score
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

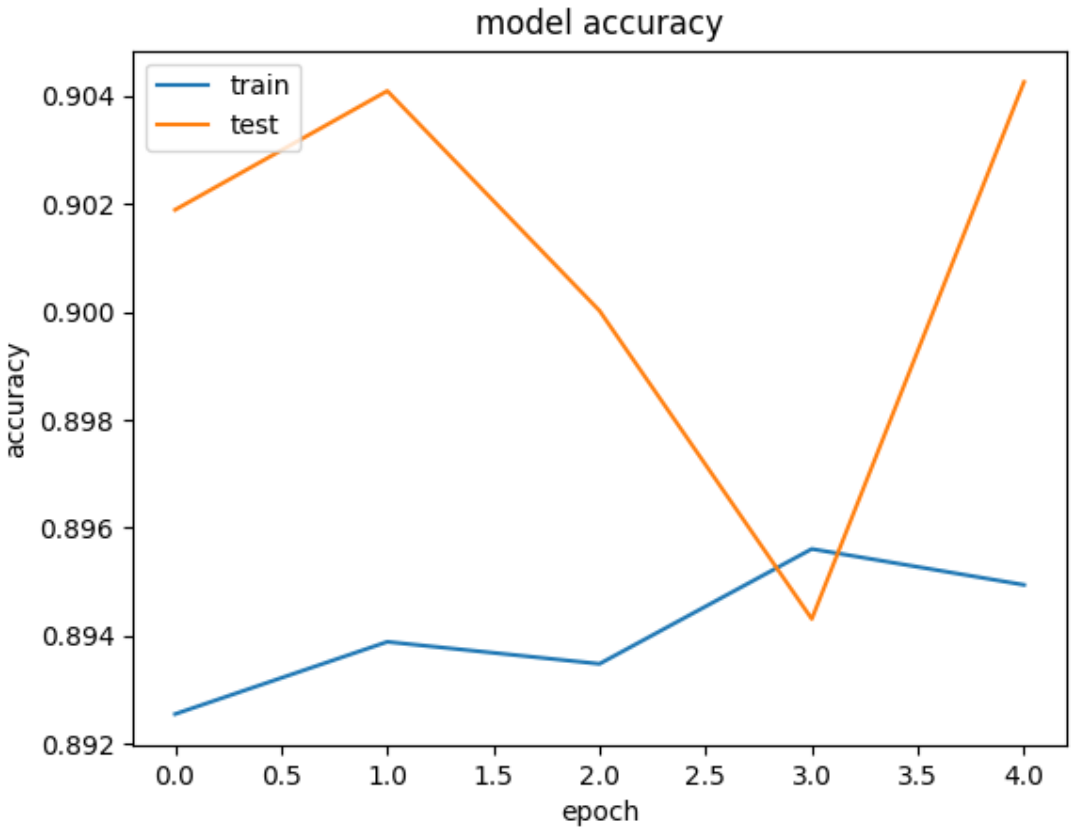

Train F1 Score Over Folds
Train F1 Score
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

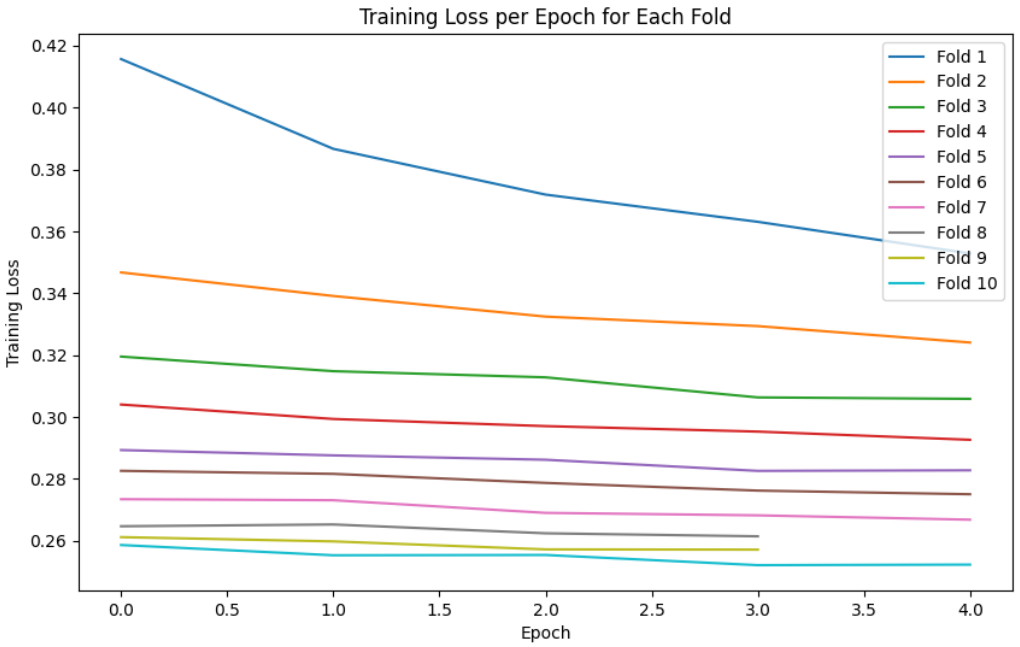


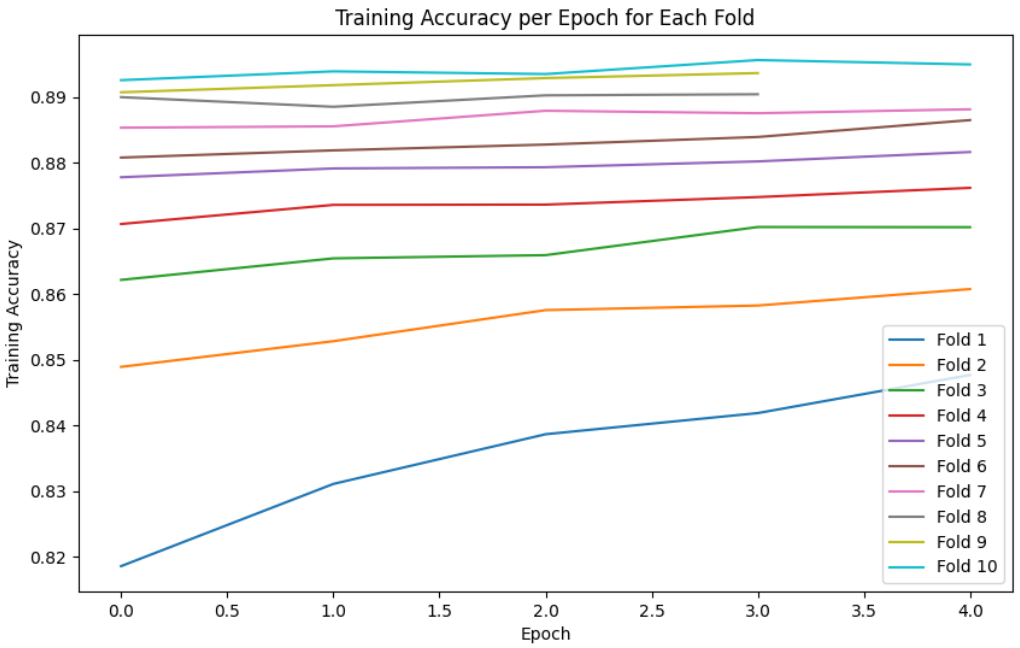
7.4 - Embeddings Estáticos, Modelo FastText + Fine Tuning:

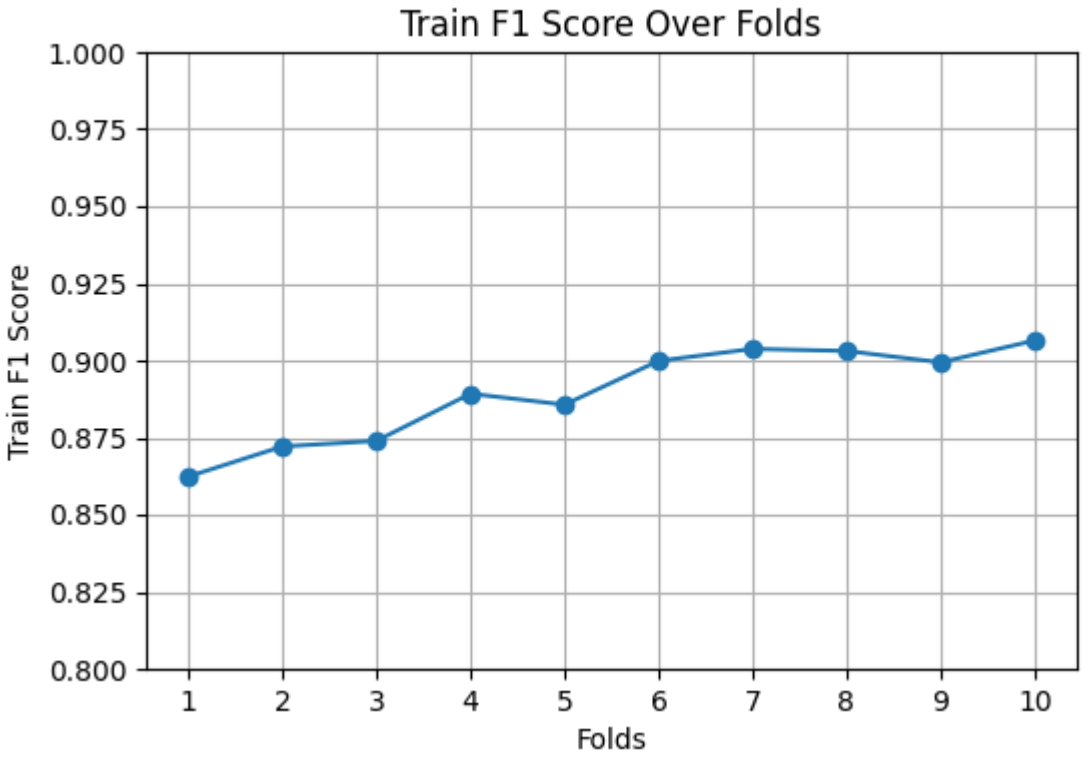
Modelo: Fast Text + Fine Tuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 5e-3	0.88 ± 0.01	0.87 ± 0.01	0.88 ± 0.01	0.88 ± 0.01

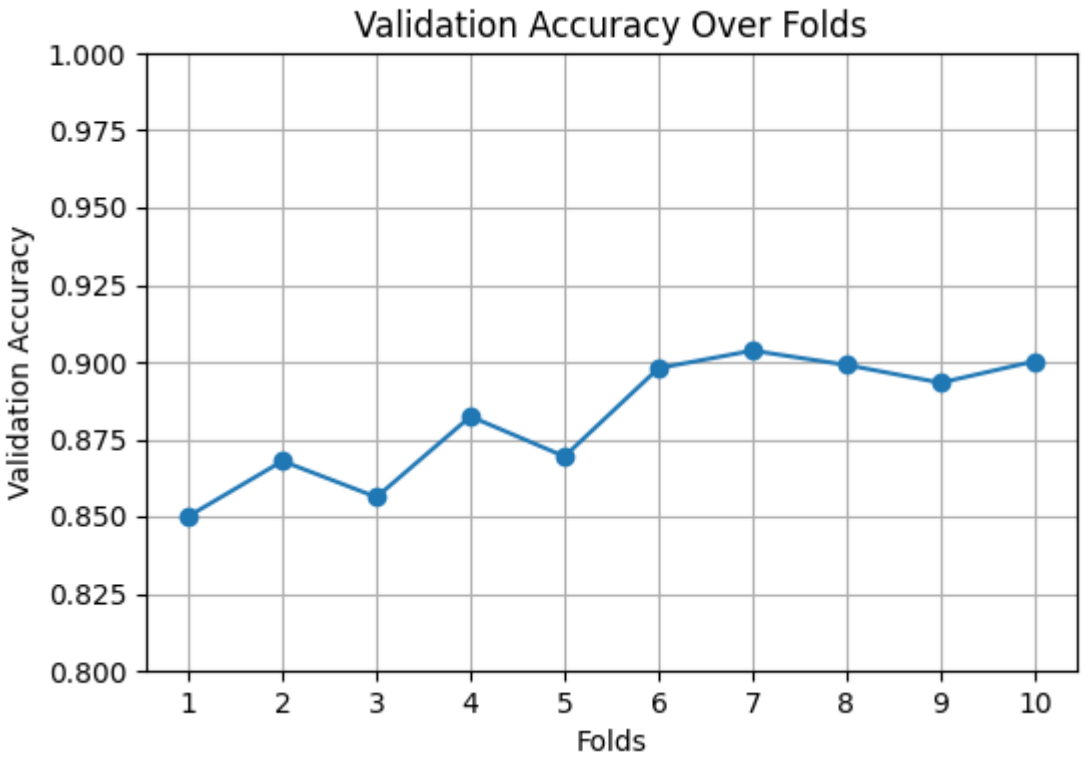

model loss
train
test
0.255
0.250
0.245
0.240
0.235
loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch

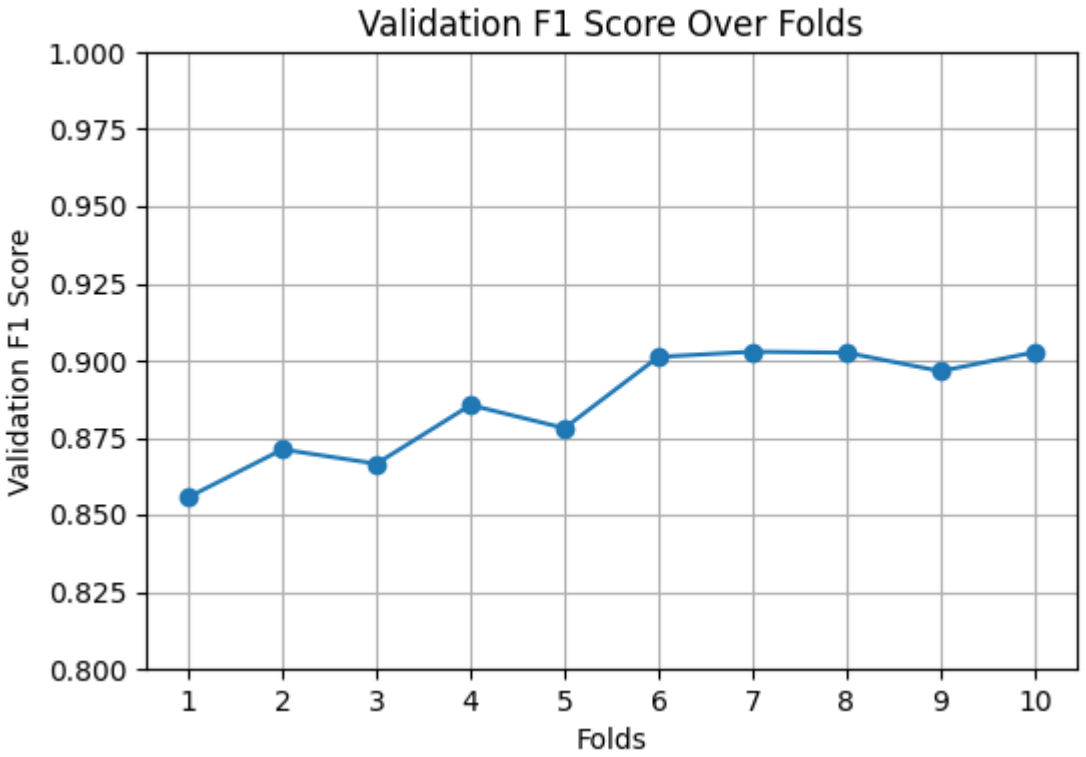

model accuracy
train
test
0.904
0.902
0.900
0.898
0.896
0.894
0.892
accuracy
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


Training Loss per Epoch for Each Fold
Training Loss
Epoch
0.42
0.40
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10


Training Accuracy per Epoch for Each Fold
Training Accuracy
Epoch
0.89
0.88
0.87
0.86
0.85
0.84
0.83
0.82
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

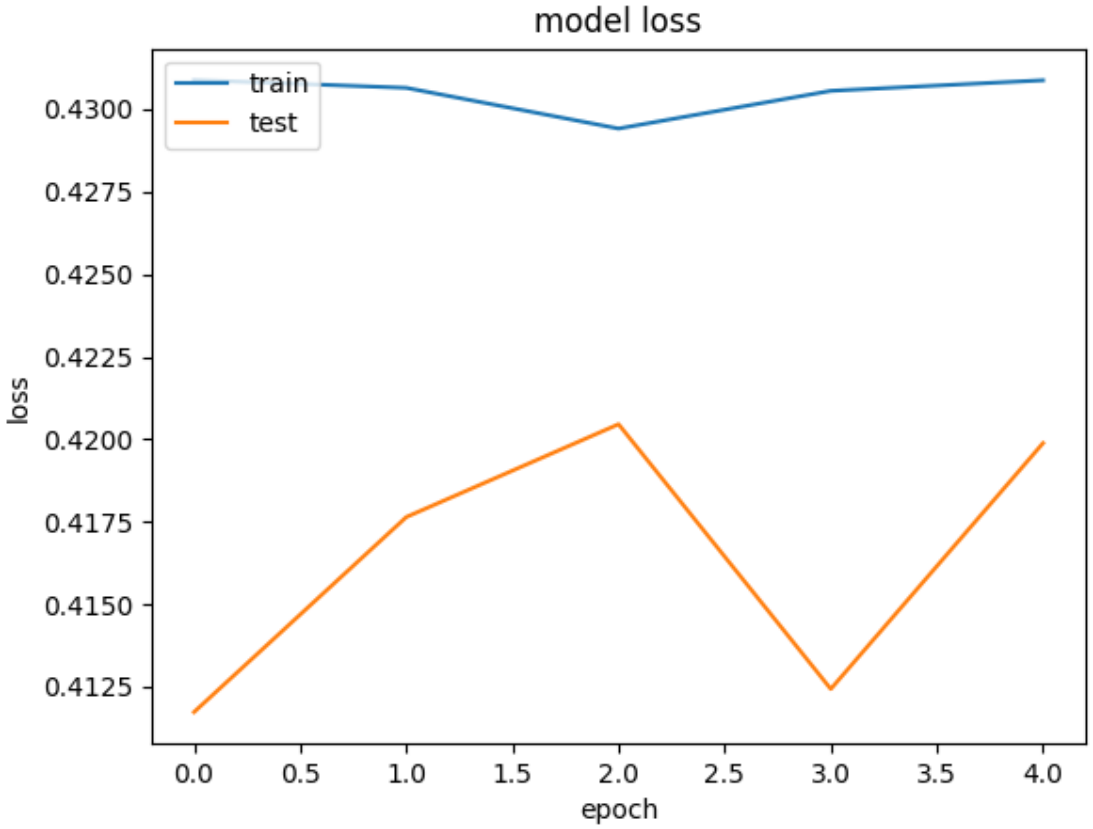

Train F1 Score Over Folds
Train F1 Score
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

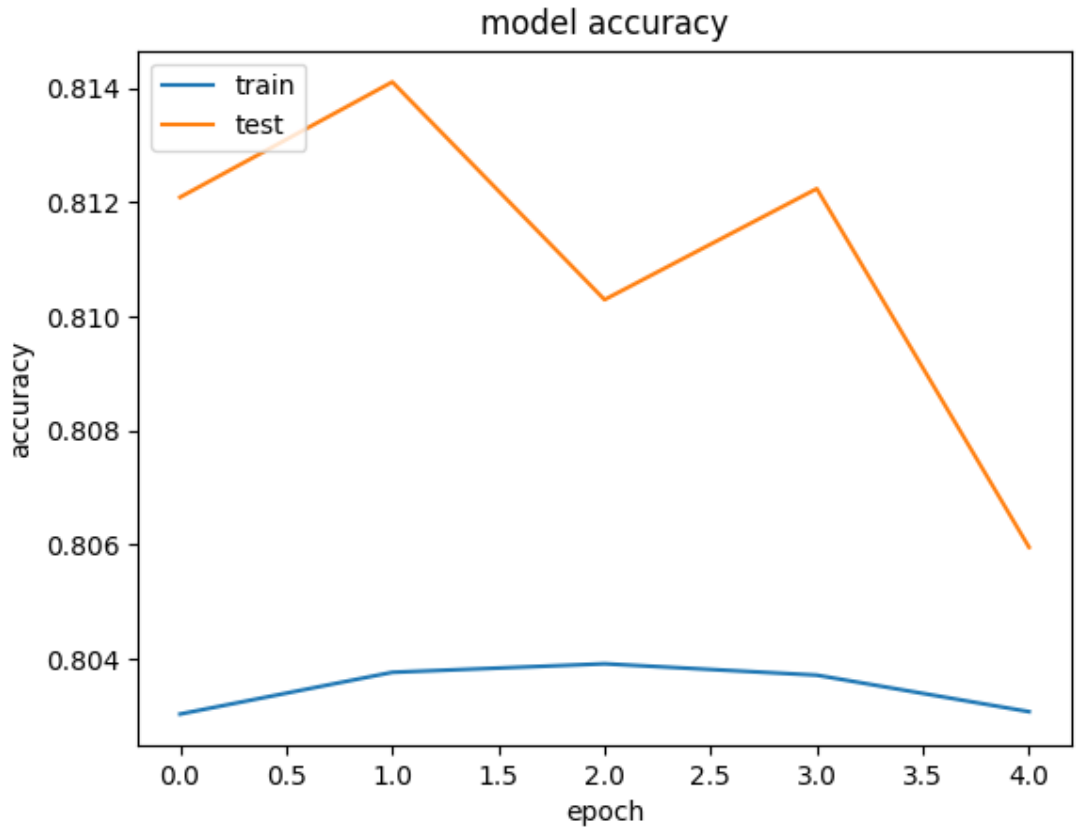

Validation Accuracy Over Folds
Validation Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

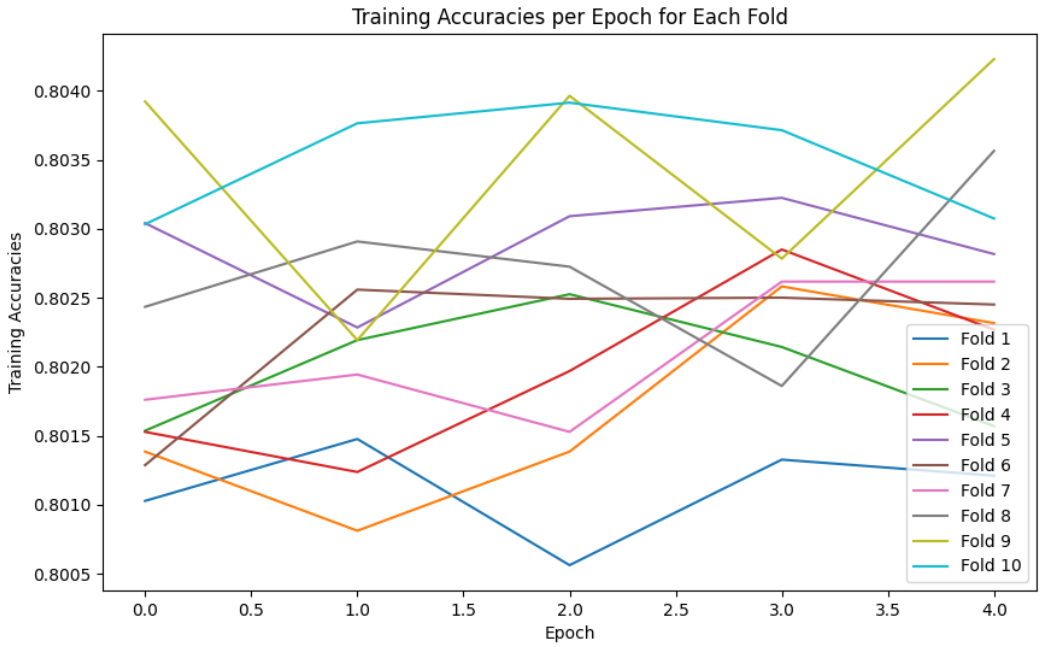


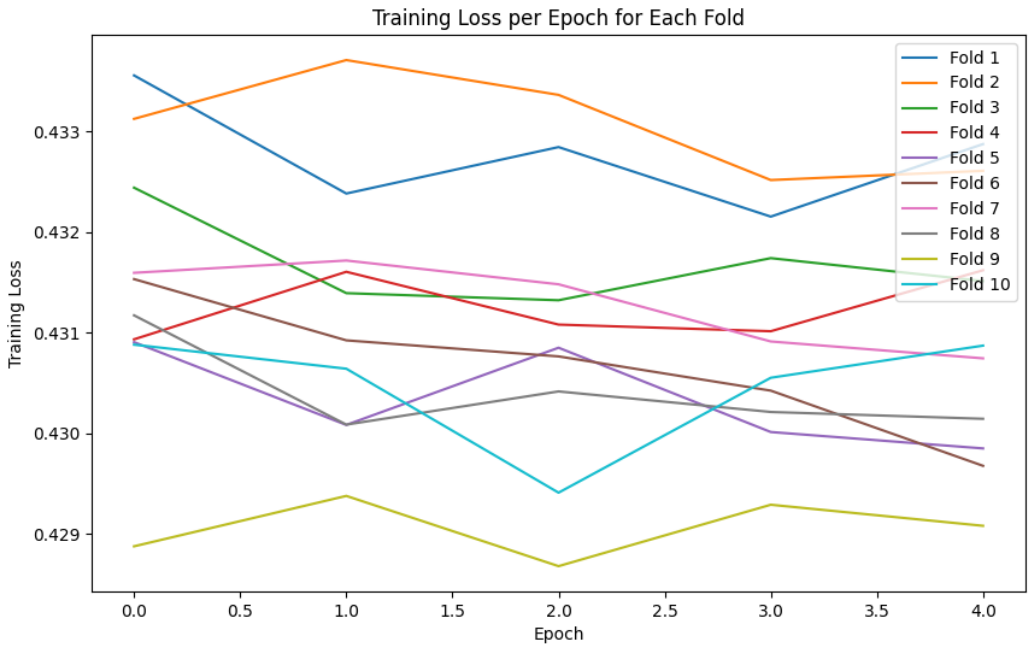
7.5 - Embeddings Estáticos, Modelo GloVe + Fine Tuning:

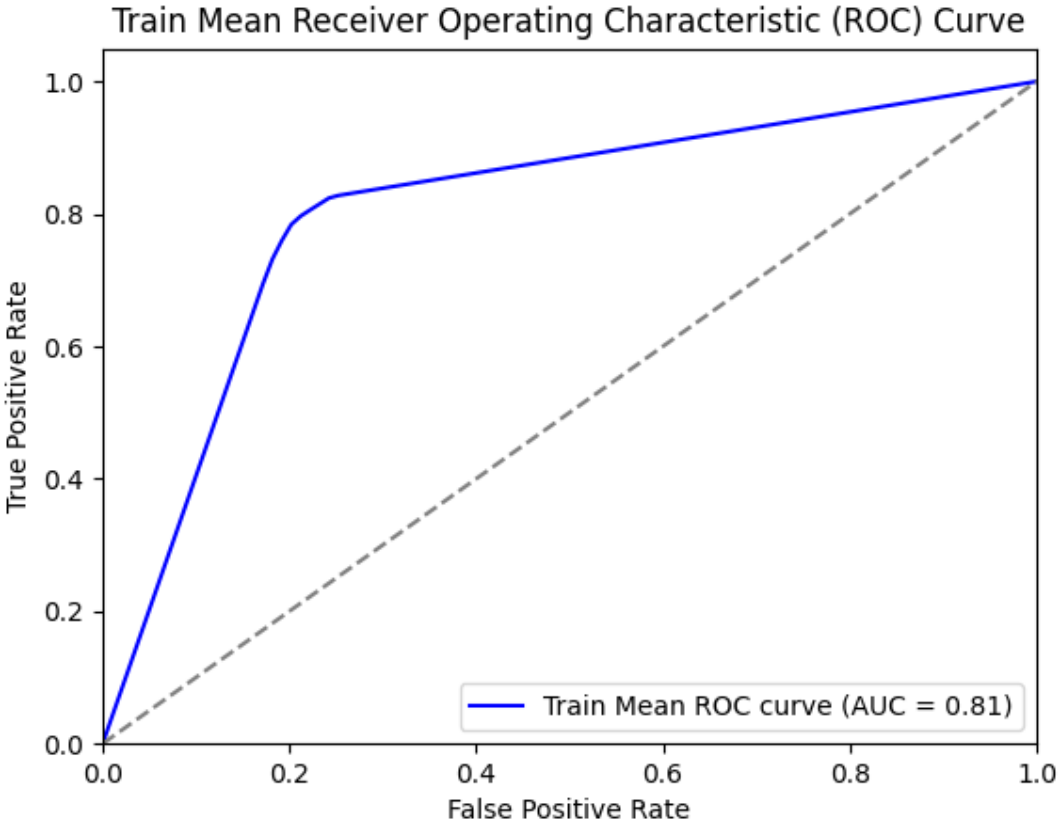
Modelo: GloVe + Fine Tuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 1e-3	0.81 ± 0.01	0.80 ± 0.01	0.80 ± 0.01	0.80 ± 0.01

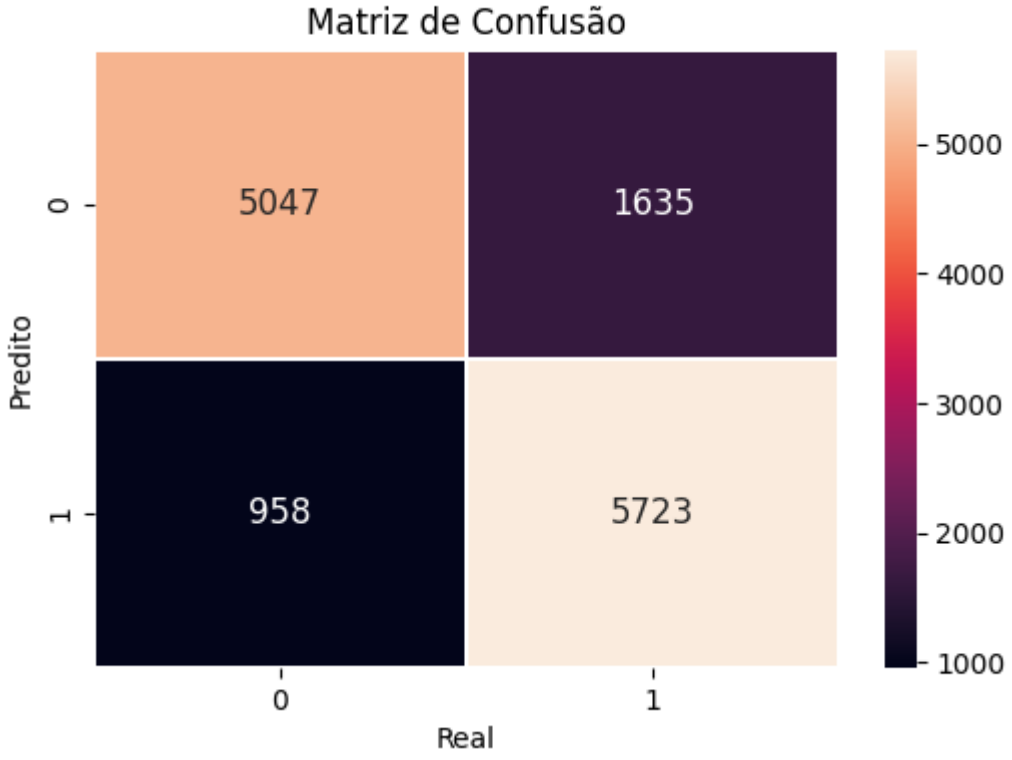

model loss
train
test
loss
0.4300
0.4275
0.4250
0.4225
0.4200
0.4175
0.4150
0.4125
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


model accuracy
train
test
accuracy
0.814
0.812
0.810
0.808
0.806
0.804
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


Training Accuracies per Epoch for Each Fold
Training Accuracies
Epoch
0.8040
0.8035
0.8030
0.8025
0.8020
0.8015
0.8010
0.8005
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10


Training Loss per Epoch for Each Fold
Training Loss
Epoch
0.433
0.432
0.431
0.430
0.429
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10


Train Mean Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
Train Mean ROC curve (AUC = 0.81)
0.0
0.2
0.4
0.6
0.8
1.0


Matriz de Confusão
Predito
Real
5047
1635
958
5723
0
1
5000
4000
3000
2000
1000

7.6 - Transformers, Modelo TLM Pequeno + FB1 + MLP:

Modelo: TLM Pequeno + FB1 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0, Learning rate = 2.5e-5	0.86 ± 0.01	0.86 ± 0.01	0.86 ± 0.01	0.86 ± 0.01

7.7 - Transformers, Modelo TLM Pequeno + FB2 + MLP:

Modelo: TLM Pequeno + FB2 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	0.85 ± 0.01	0.85 ± 0.01	0.87 ± 0.01	0.87 ± 0.01

7.8 - Transformers, Modelo TLM Grande + FB1 + MLP:

Modelo: TLM Grande + FB1 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	0.88 ± 0.01	0.87 ± 0.01	0.88 ± 0.01	0.87 ± 0.01

7.9 - Transformers, Modelo TLM Grande + FB2 + MLP:

Modelo: TLM Grande + FB2 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0, Learning rate = 2.5e-5	0.91 ± 0.01	0.90 ± 0.02	0.91 ± 0.01	0.90 ± 0.02

7.10 - Transformers, Modelo TLM Pequeno + Fine Tuning:

Modelo: TLM Pequeno + FineTuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	0.90 ± 0.00	0.90 ± 0.01	0.90 ± 0.00	0.90 ± 0.01

7.11 - Transformers, Modelo TLM Grande + Fine Tuning:

Modelo: TLM Grande + FineTuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	0.89 ± 0.00	0.88 ± 0.00	0.89 ± 0.00	0.88 ± 0.00

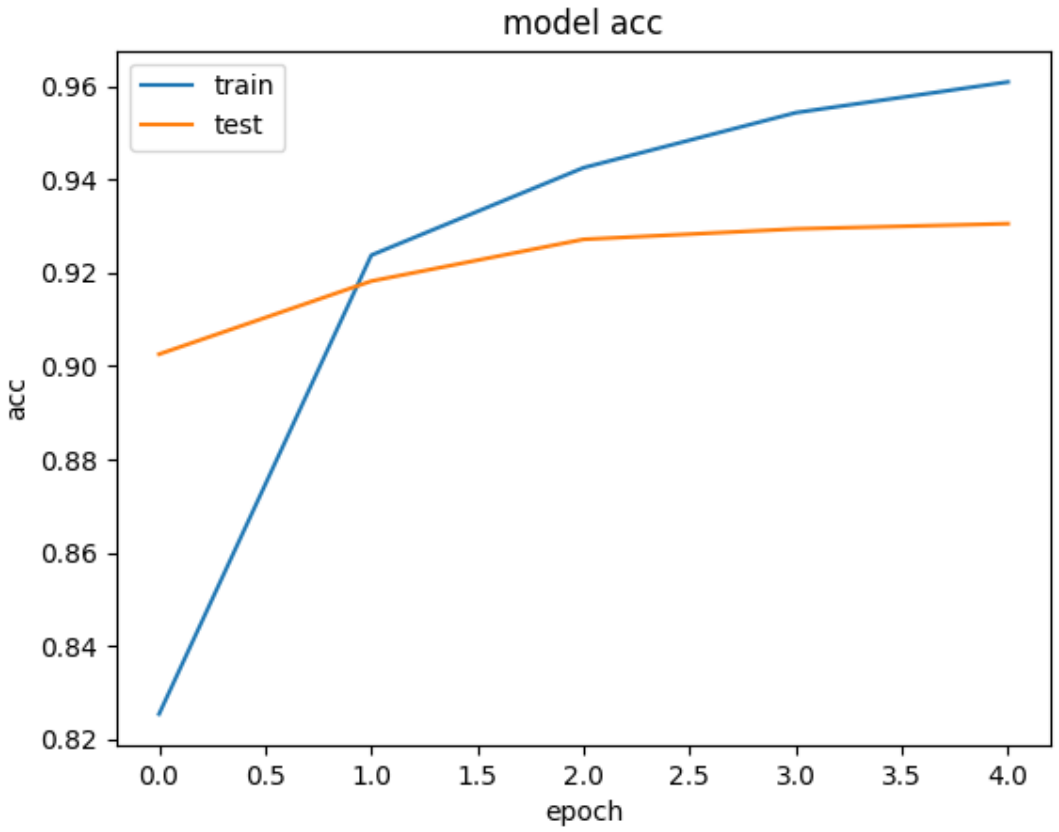
7.12 - Transformers, Modelo TLM Pequeno Multilingue:

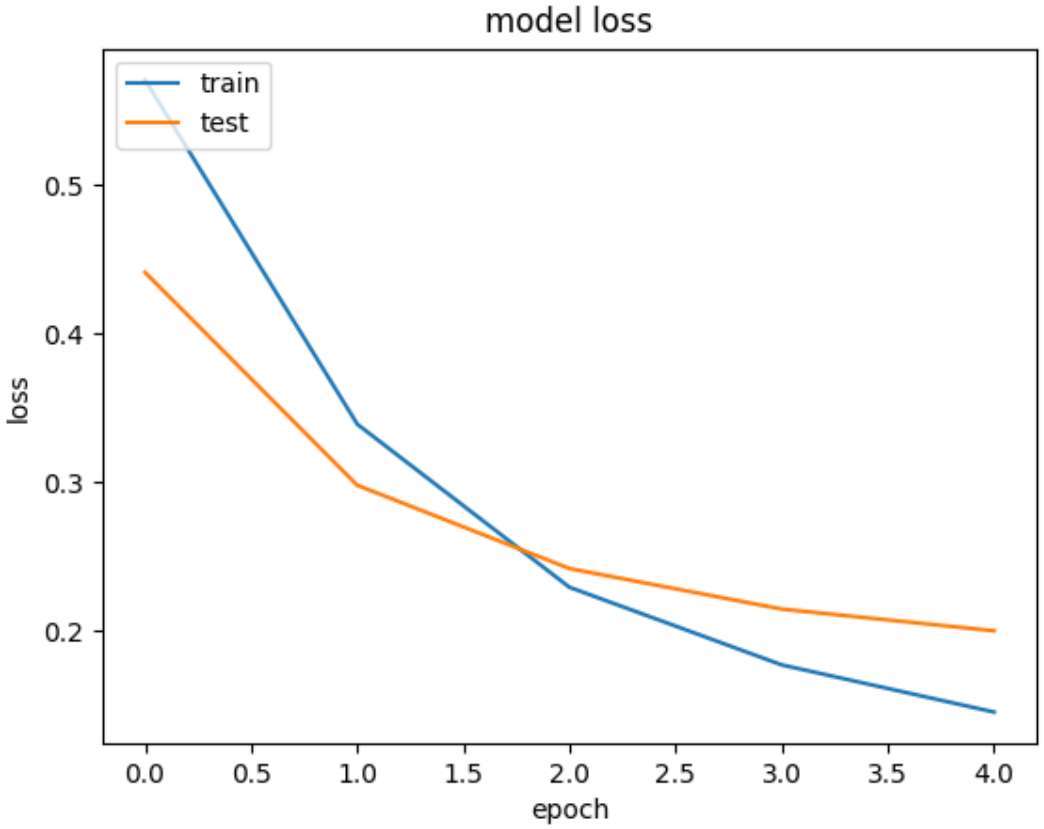
Modelo: TLM Pequeno Multilíngue				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	0.85 ± 0.01	0.84 ± 0.01	0.85 ± 0.01	0.84 ± 0.01

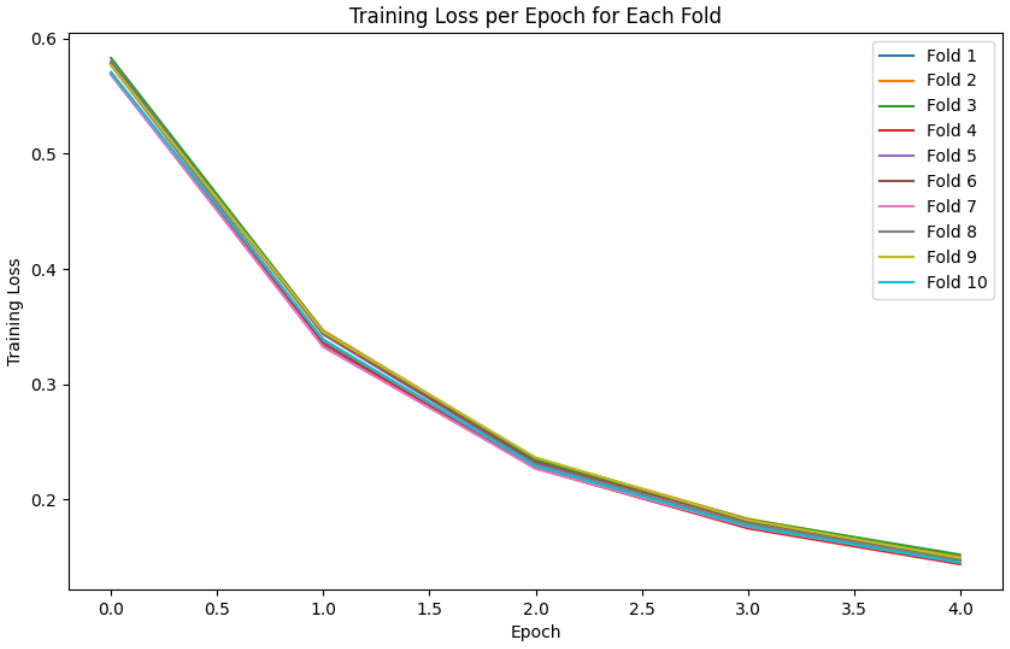
8 - Análises Estatísticas das bases - Melhores Resultados - Base B2W:

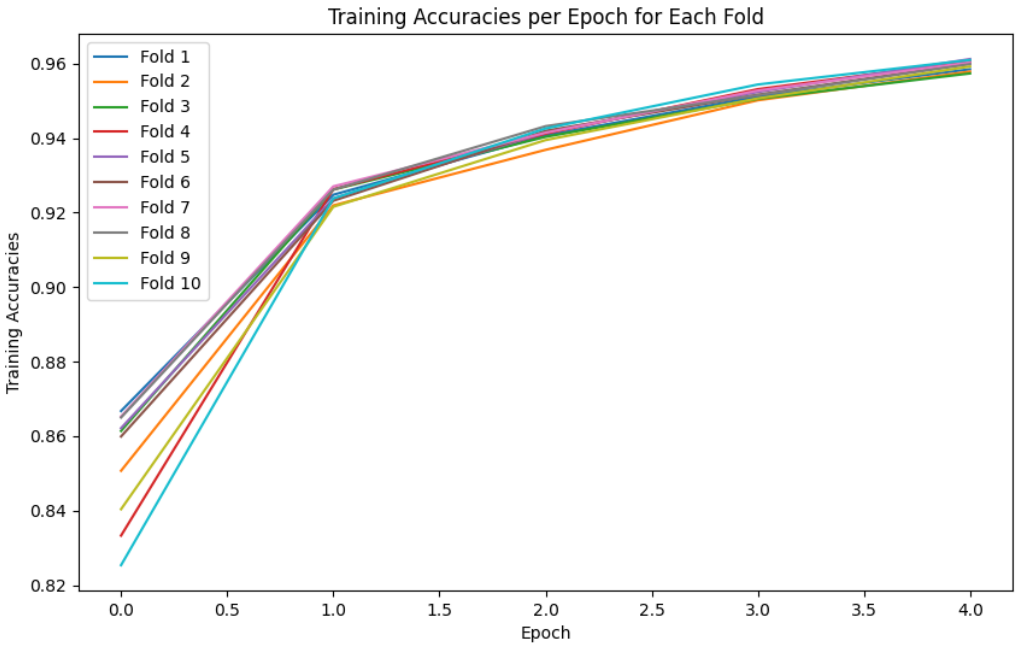
8.1 - Embeddings Estáticos, Modelo TFIDF + MLP:

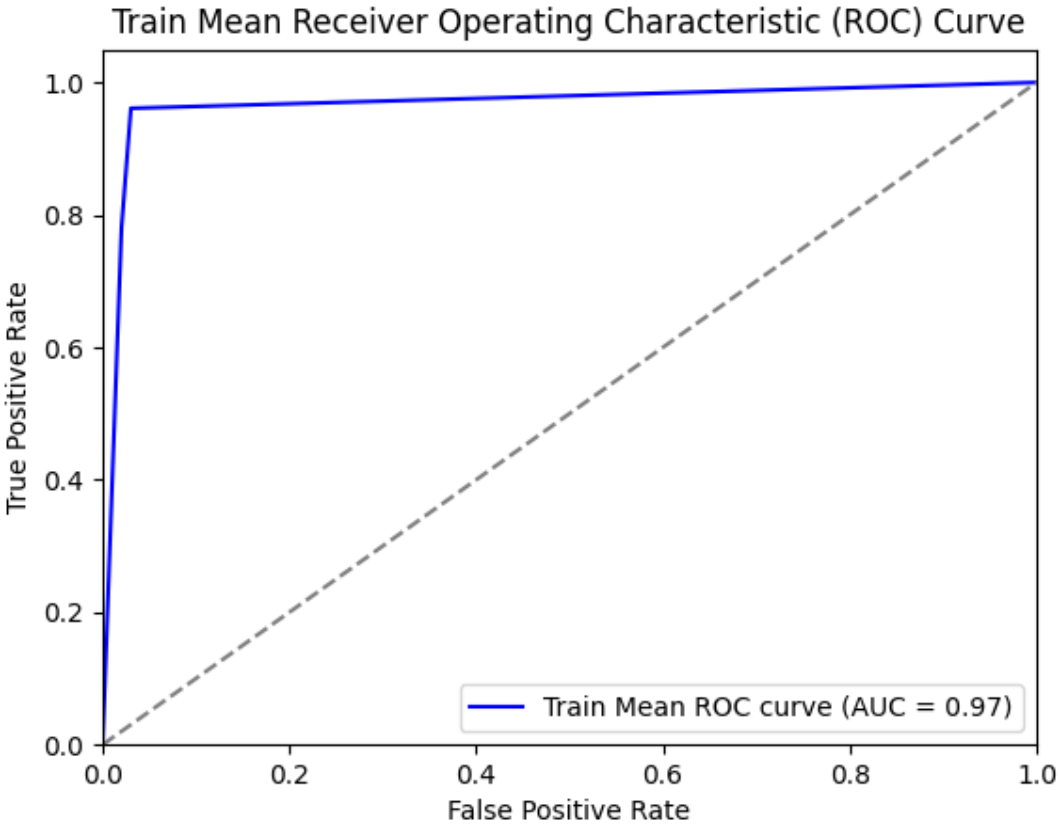
Modelo: TFIDF + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 5e-4	0.96 ± 0.00	0.94 ± 0.01	0.93 ± 0.01	0.93 ± 0.01

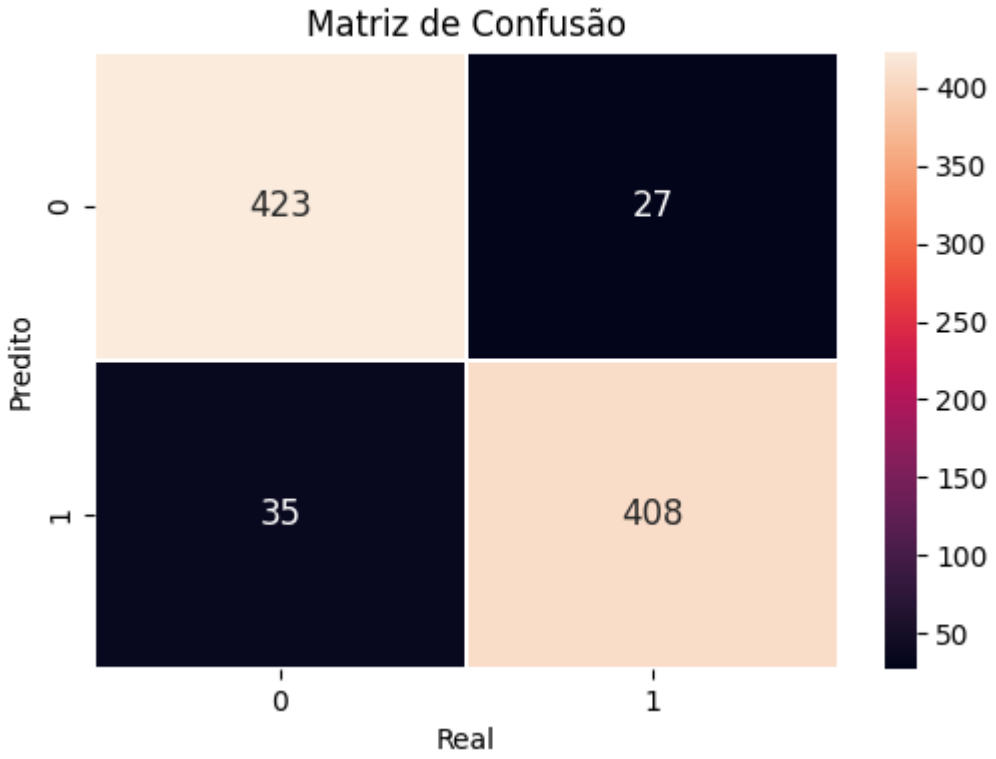



model loss
train
test
loss
epoch
0.5
0.4
0.3
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
Epoch
0.6
0.5
0.4
0.3
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

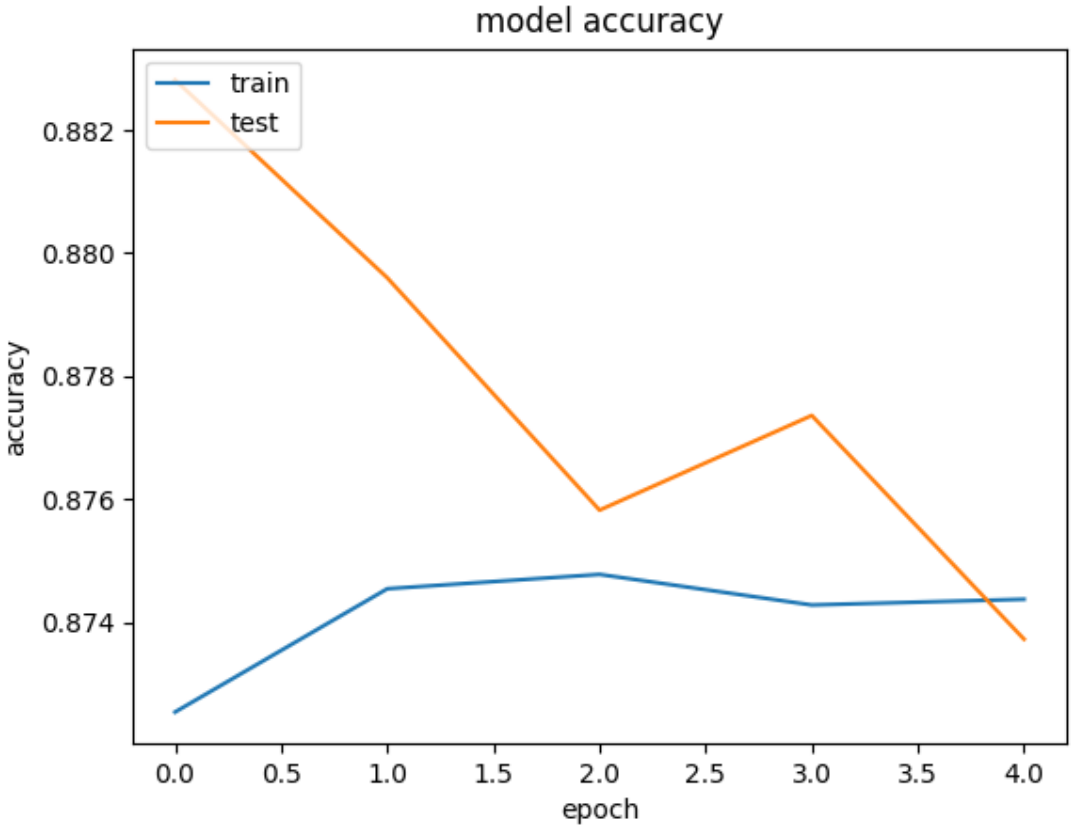

Training Accuracies per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Accuracies
Epoch
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

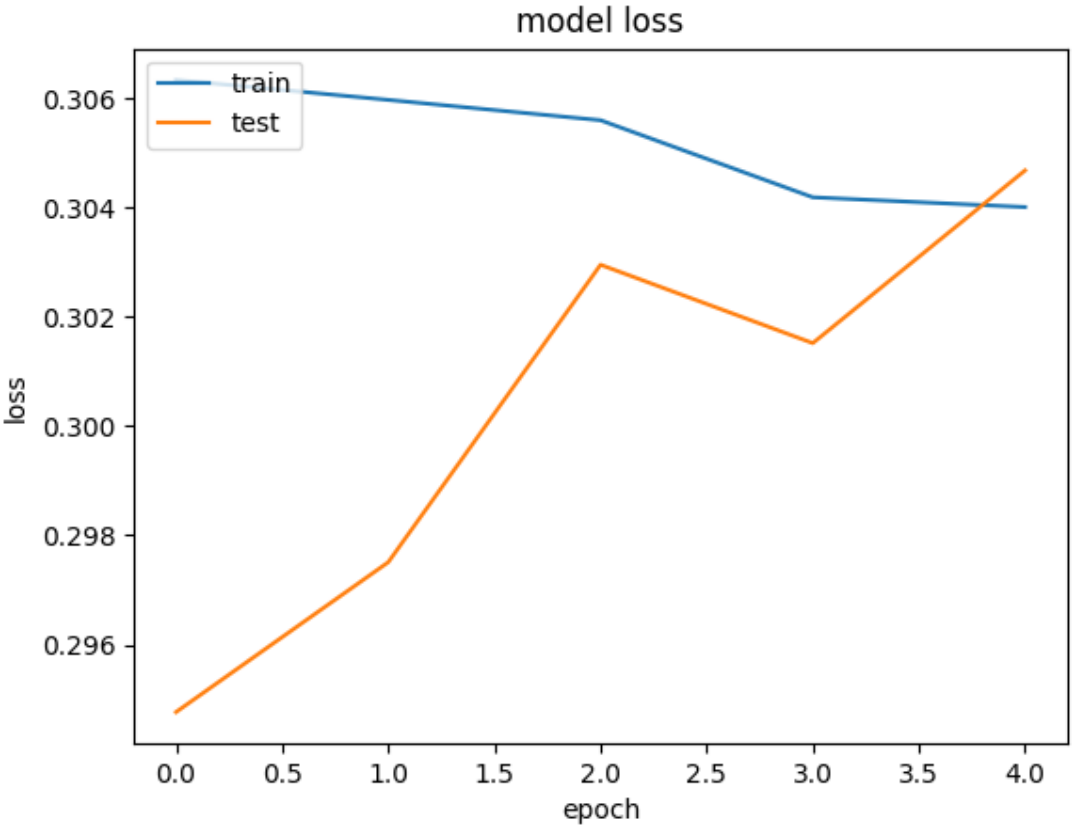

Train Mean Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
Train Mean ROC curve (AUC = 0.97)
0.0
0.2
0.4
0.6
0.8
1.0

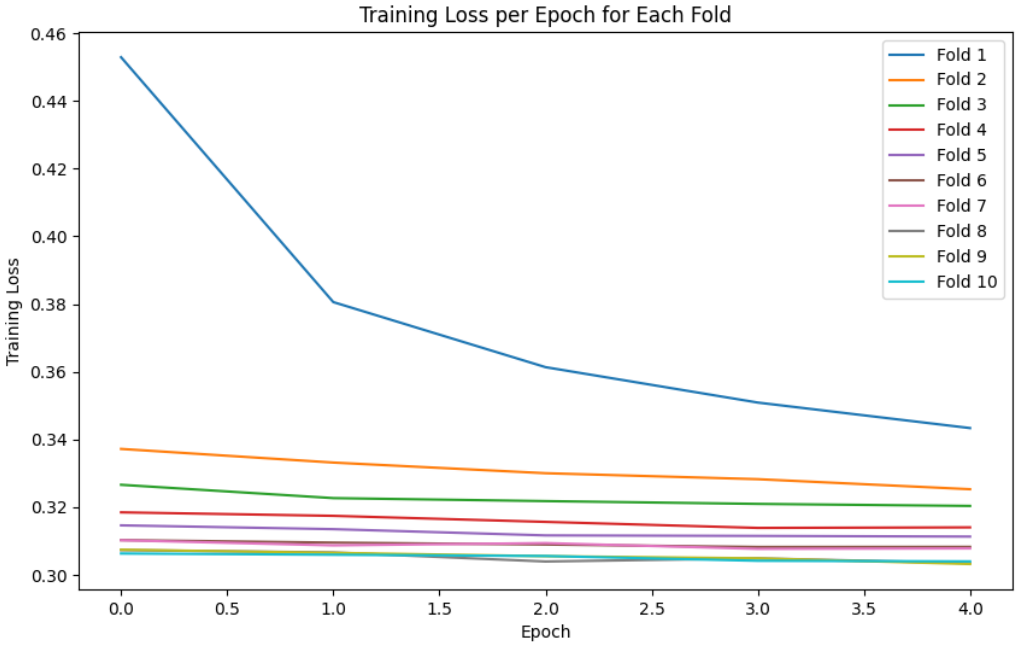


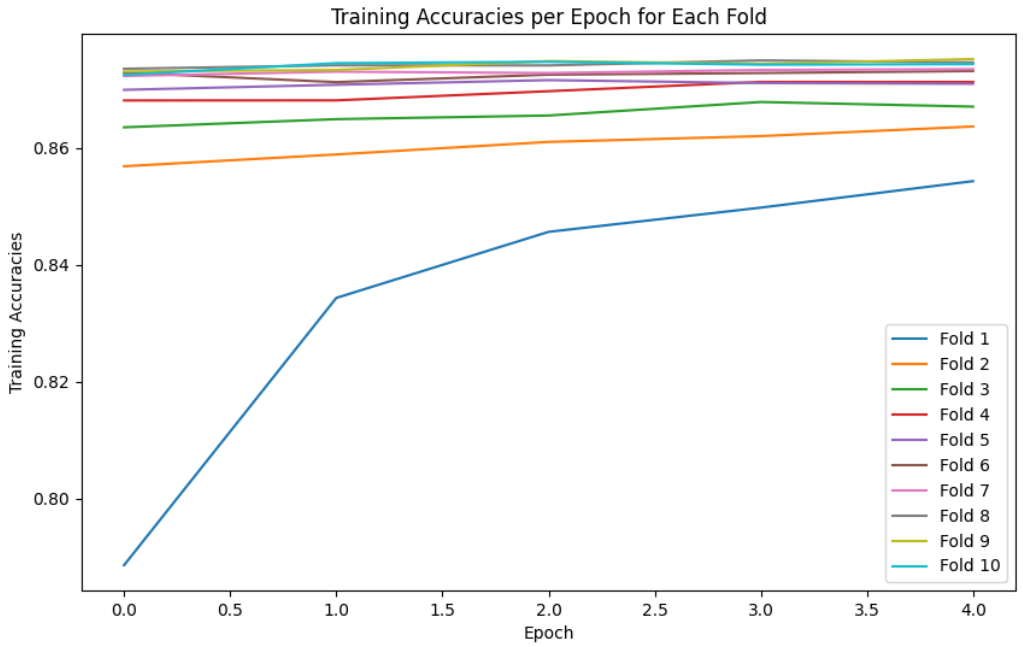
8.2 - Embeddings Estáticos, Modelo GloVe + MLP:

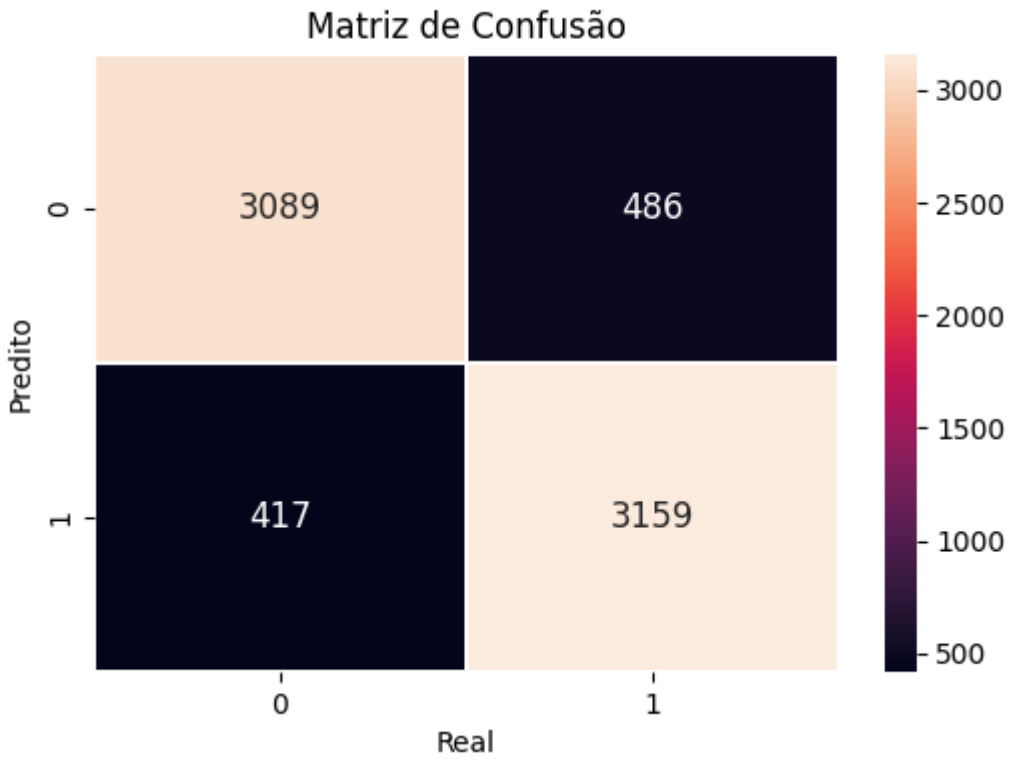
Modelo: GloVe + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 1e-3	0.87 ± 0.01	0.86 ± 0.01	0.87 ± 0.01	0.86 ± 0.01


model accuracy
train
test
0.882
0.880
0.878
0.876
0.874
accuracy
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


model loss
train
test
loss
epoch
0.306
0.304
0.302
0.300
0.298
0.296
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

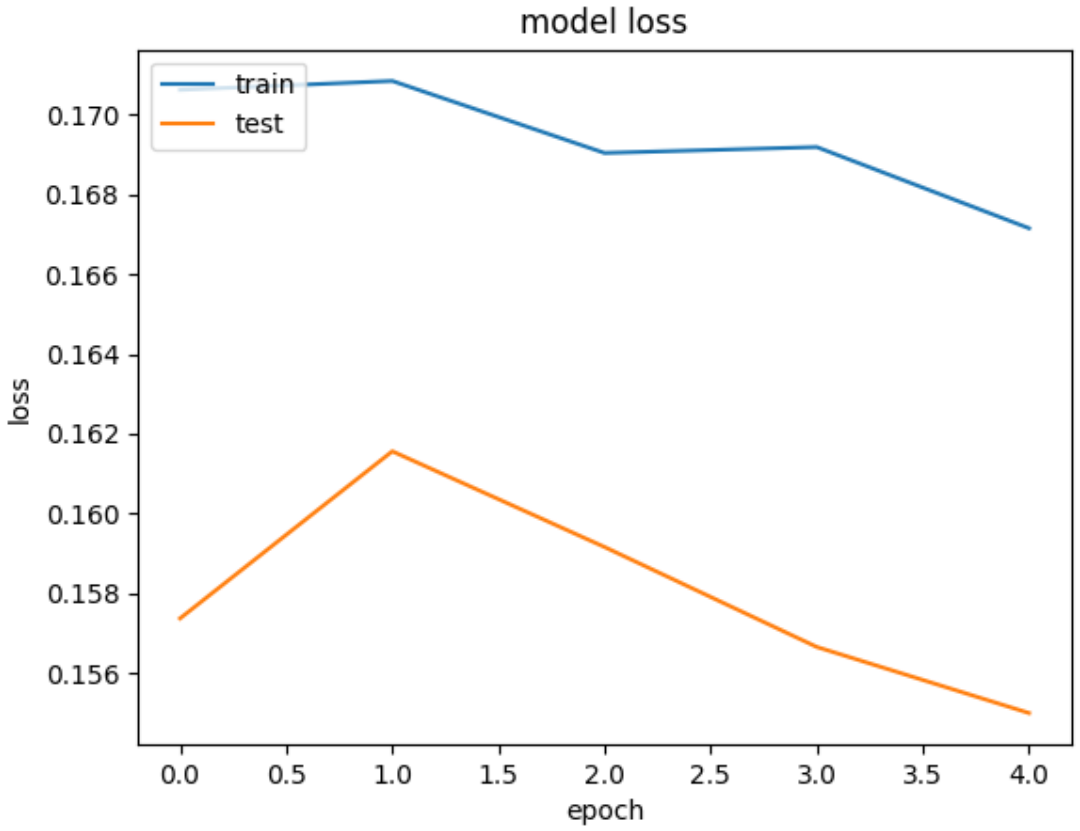

Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
Epoch
0.46
0.44
0.42
0.40
0.38
0.36
0.34
0.32
0.30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

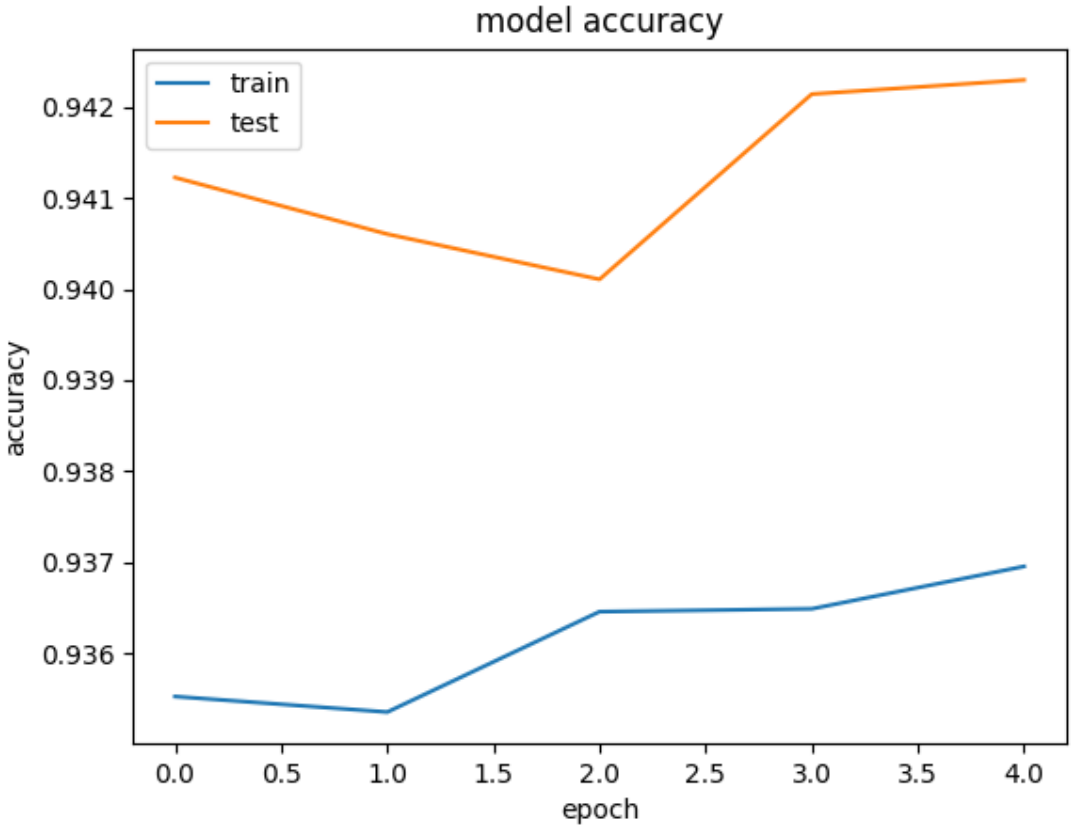

Training Accuracies per Epoch for Each Fold
Training Accuracies
Epoch
0.80
0.82
0.84
0.86
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

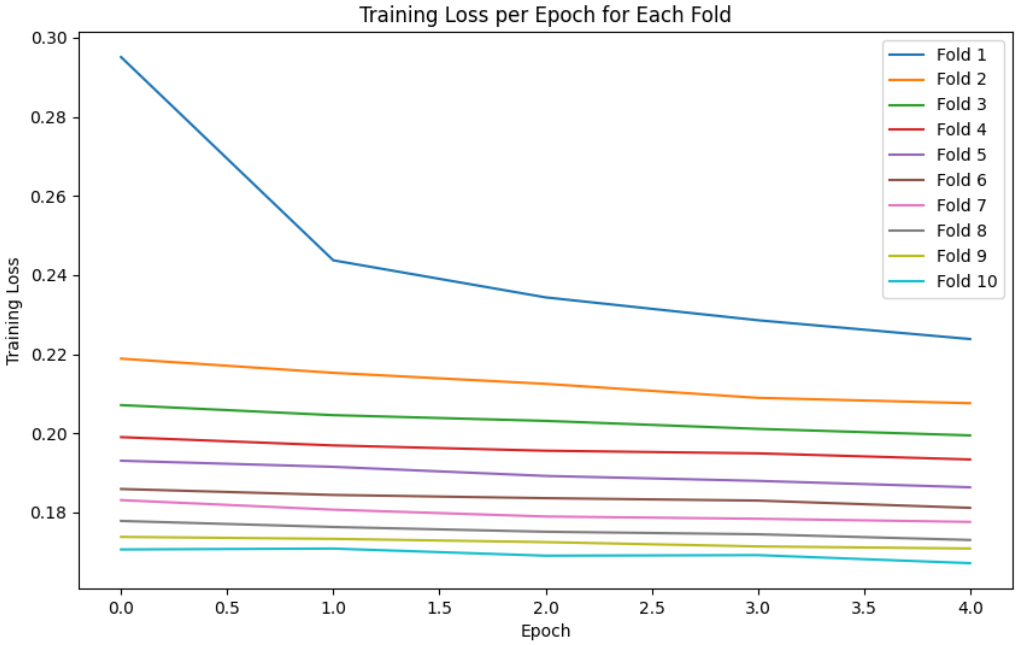

Matriz de Confusão
Predito
Real
0
1
3089
486
417
3159
3000
2500
2000
1500
1000
500

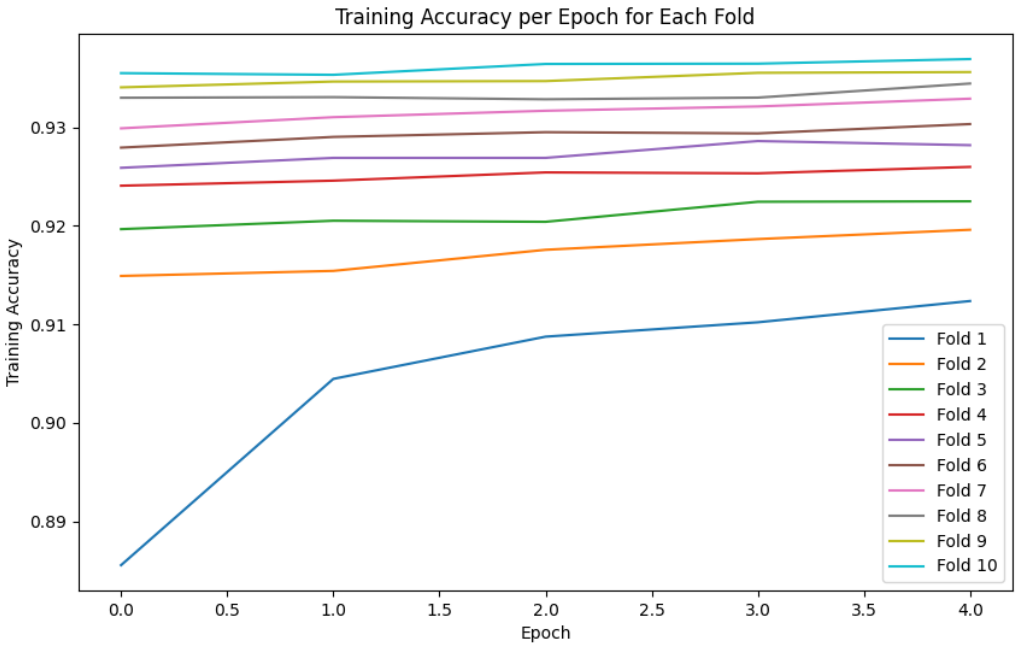
8.3 - Embeddings Estáticos, Modelo FastText + MLP:

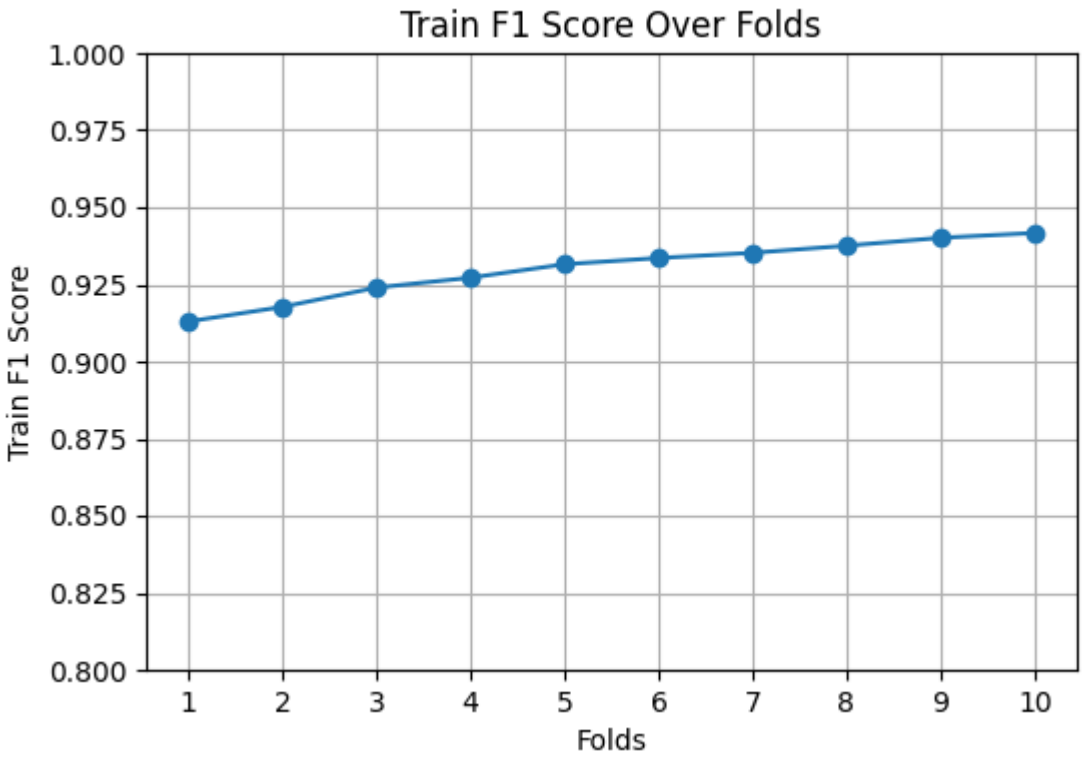
Modelo: FastText + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 1e-3	0.93 ± 0.01	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01

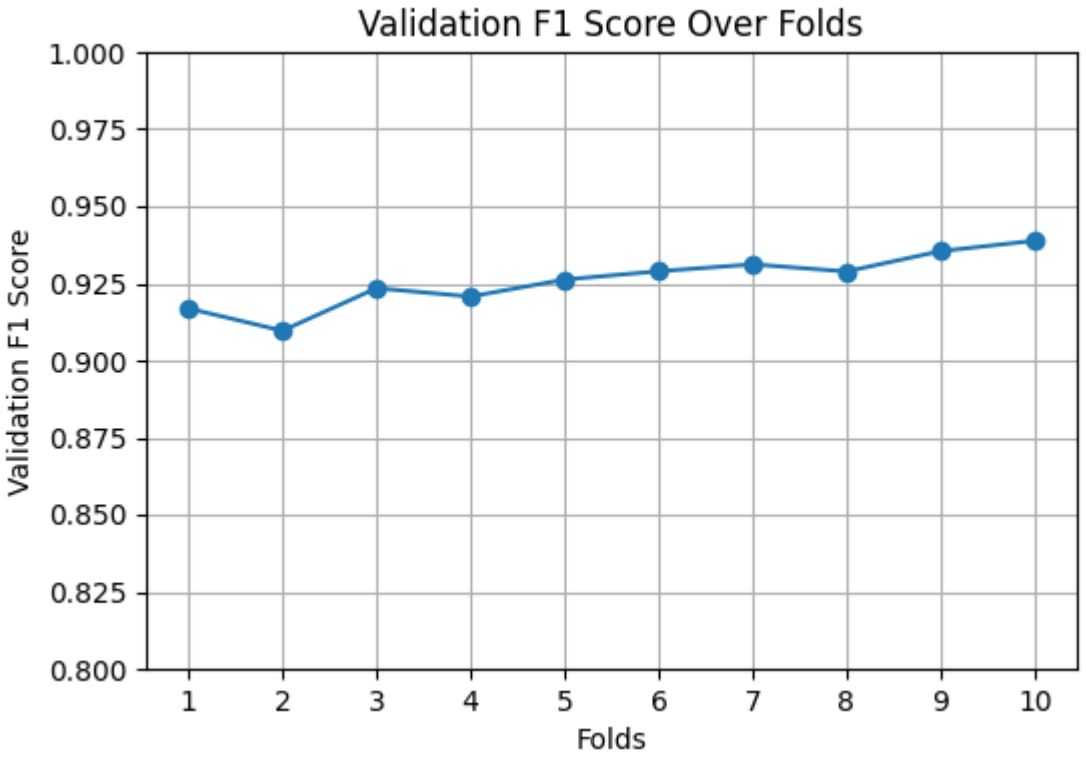


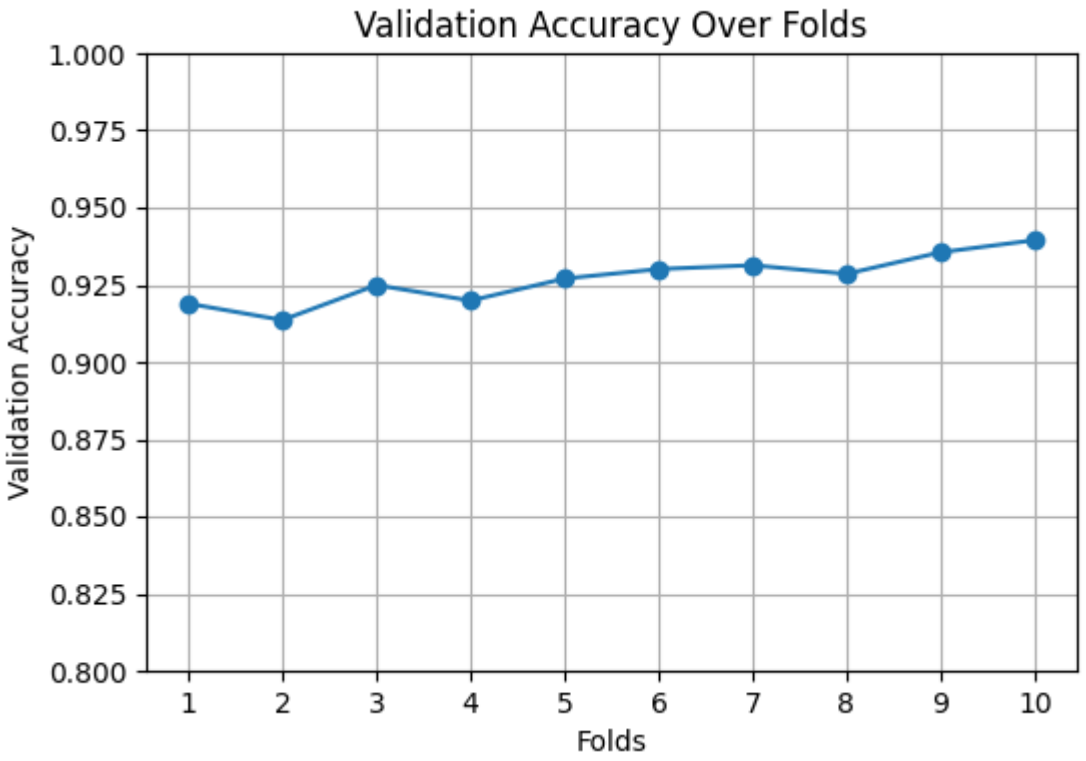

model accuracy
train
test
accuracy
epoch
0.942
0.941
0.940
0.939
0.938
0.937
0.936
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
Epoch
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Training Accuracy per Epoch for Each Fold
Training Accuracy
Epoch
0.93
0.92
0.91
0.90
0.89
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

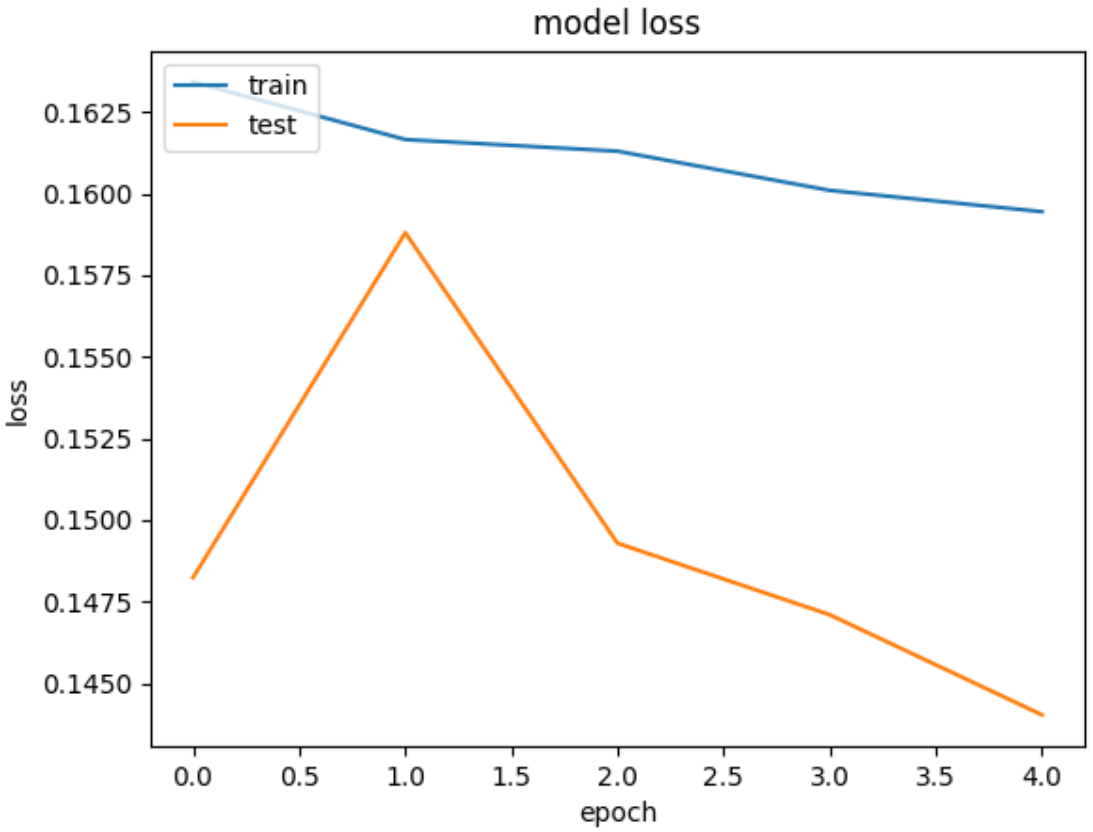

Train F1 Score Over Folds
Train F1 Score
Folds
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10

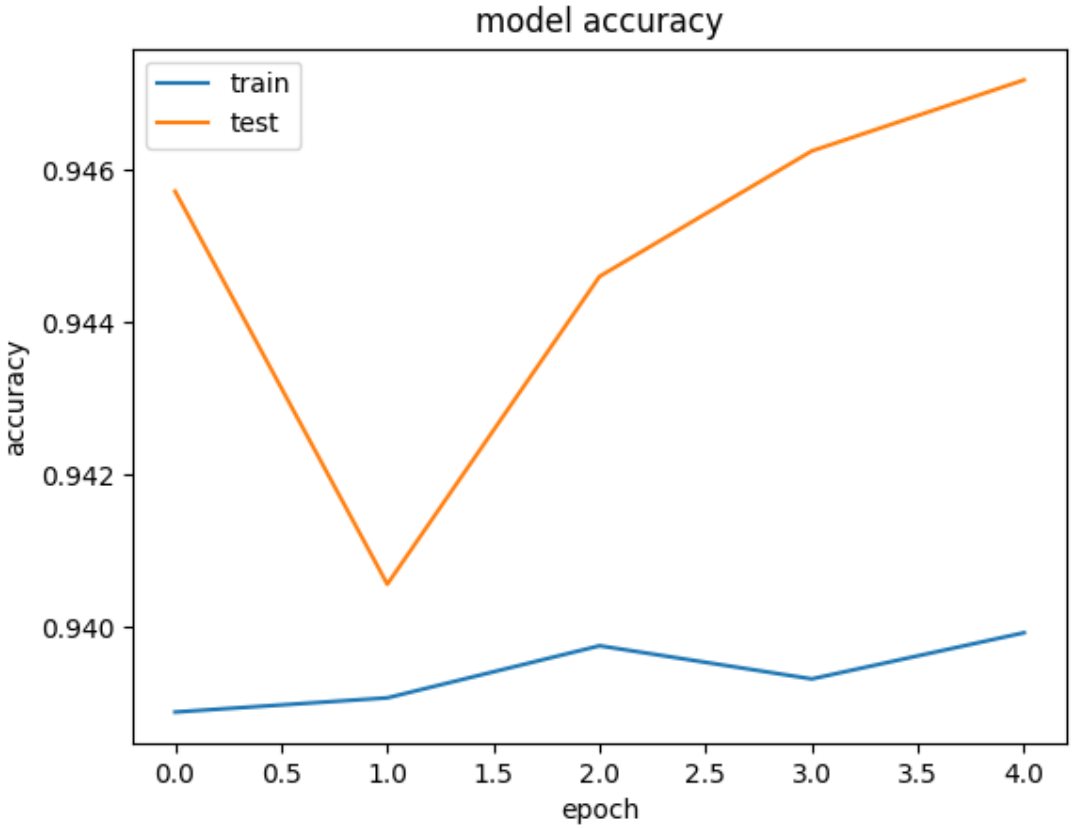

Validation F1 Score Over Folds
Validation F1 Score
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

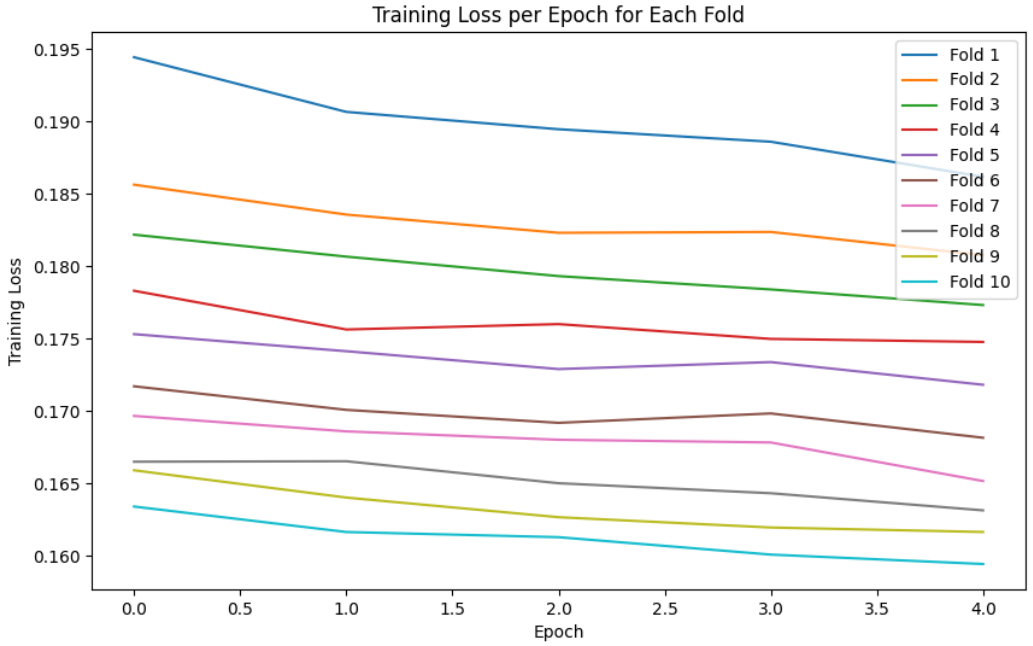

Validation Accuracy Over Folds
Validation Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

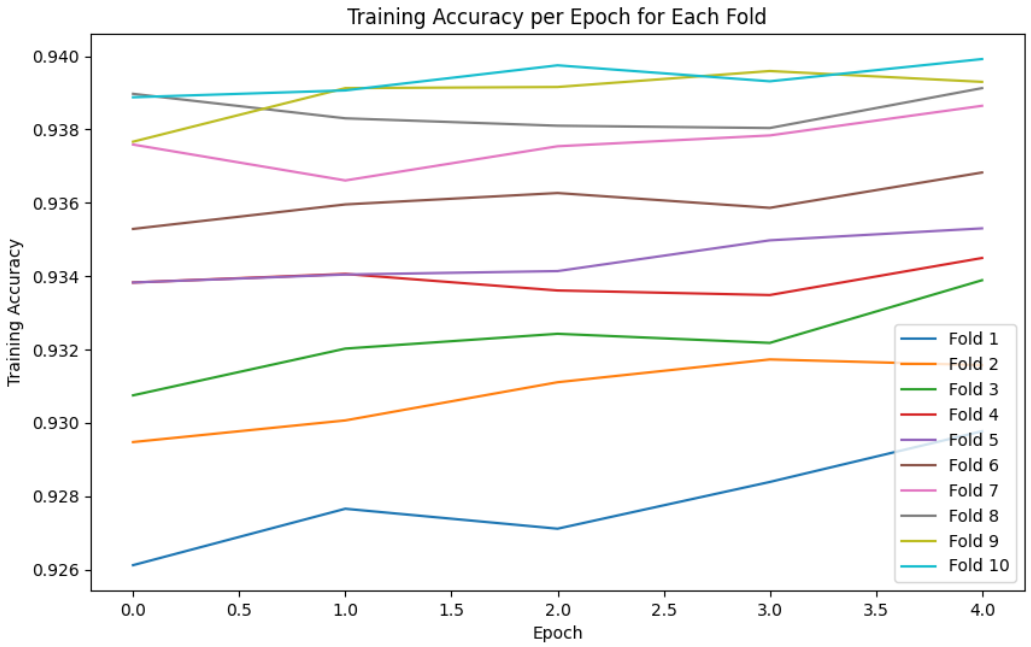
8.4 - Embeddings Estáticos, Modelo FastText + Fine Tuning:

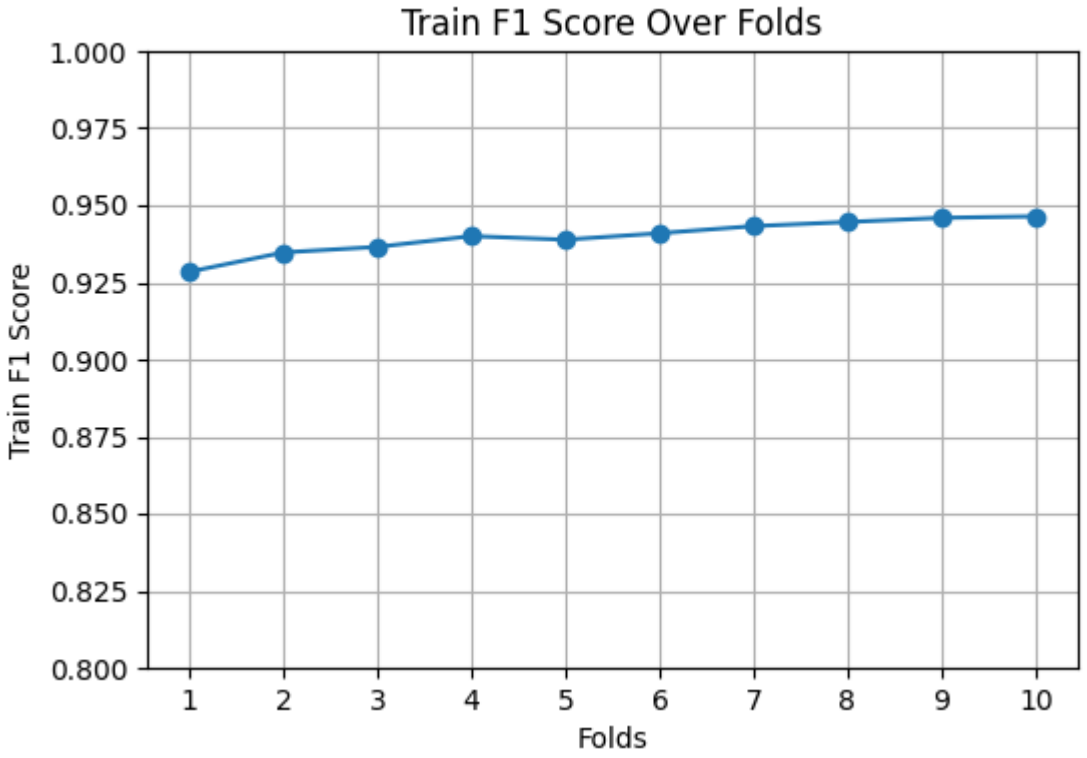
Modelo: Fast Text + Fine Tuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 1e-3	0.93 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.93 ± 0.01

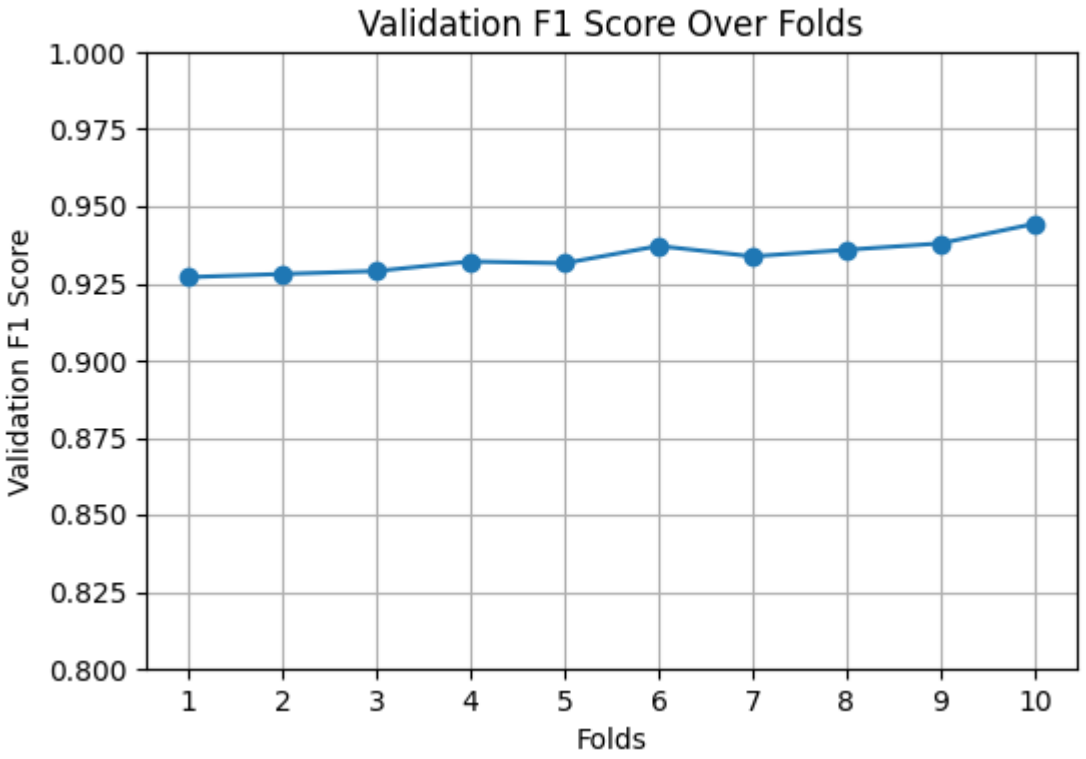


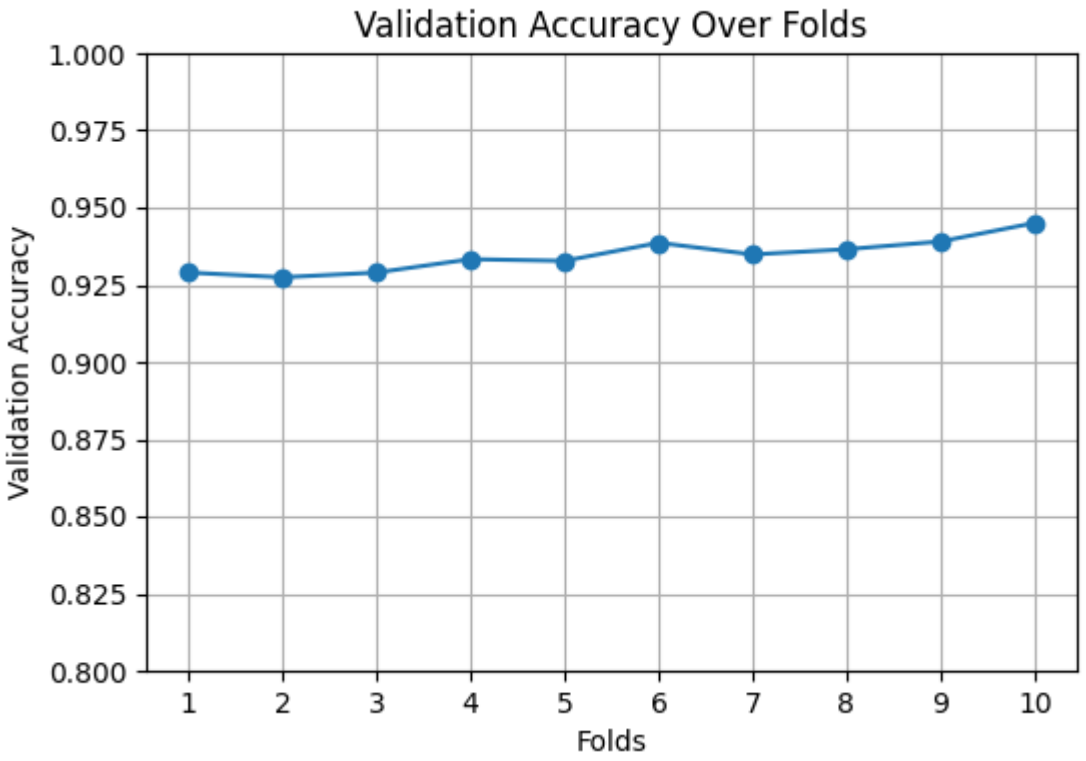

model accuracy
train
test
accuracy
0.946
0.944
0.942
0.940
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
0.195
0.190
0.185
0.180
0.175
0.170
0.165
0.160
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch


Training Accuracy per Epoch for Each Fold
Training Accuracy
Epoch
0.940
0.938
0.936
0.934
0.932
0.930
0.928
0.926
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10

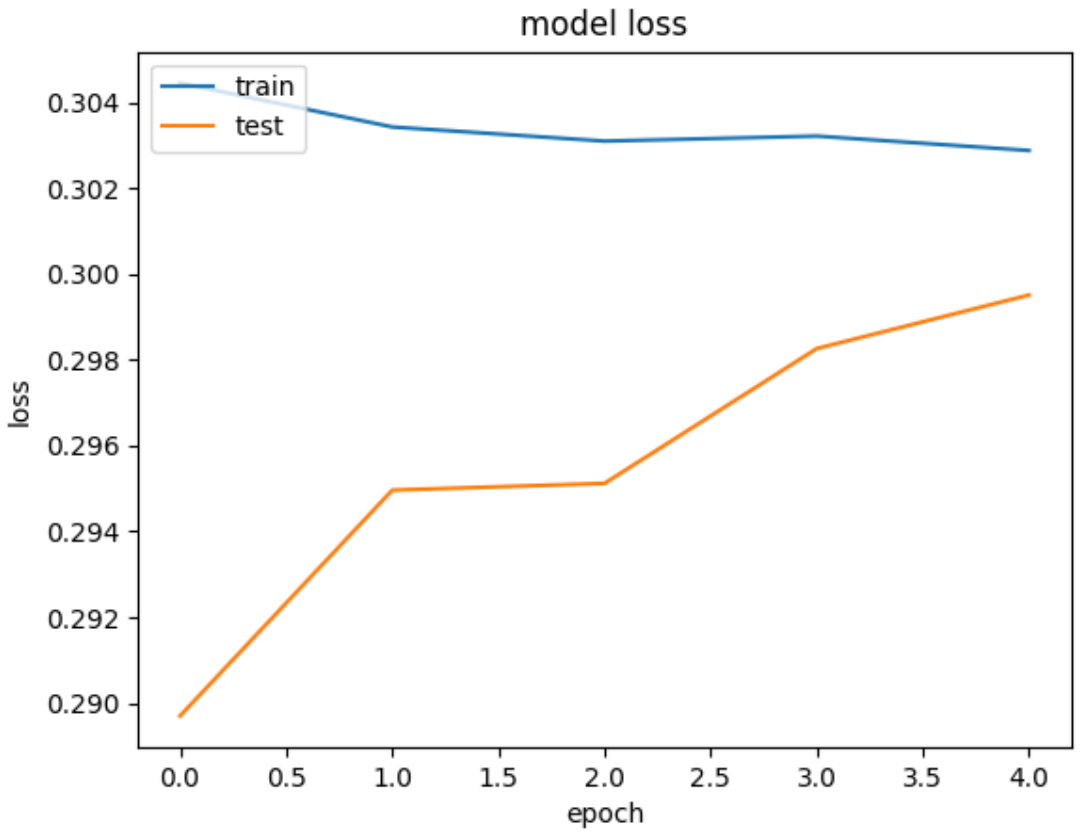

Train F1 Score Over Folds
Train F1 Score
Folds
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10

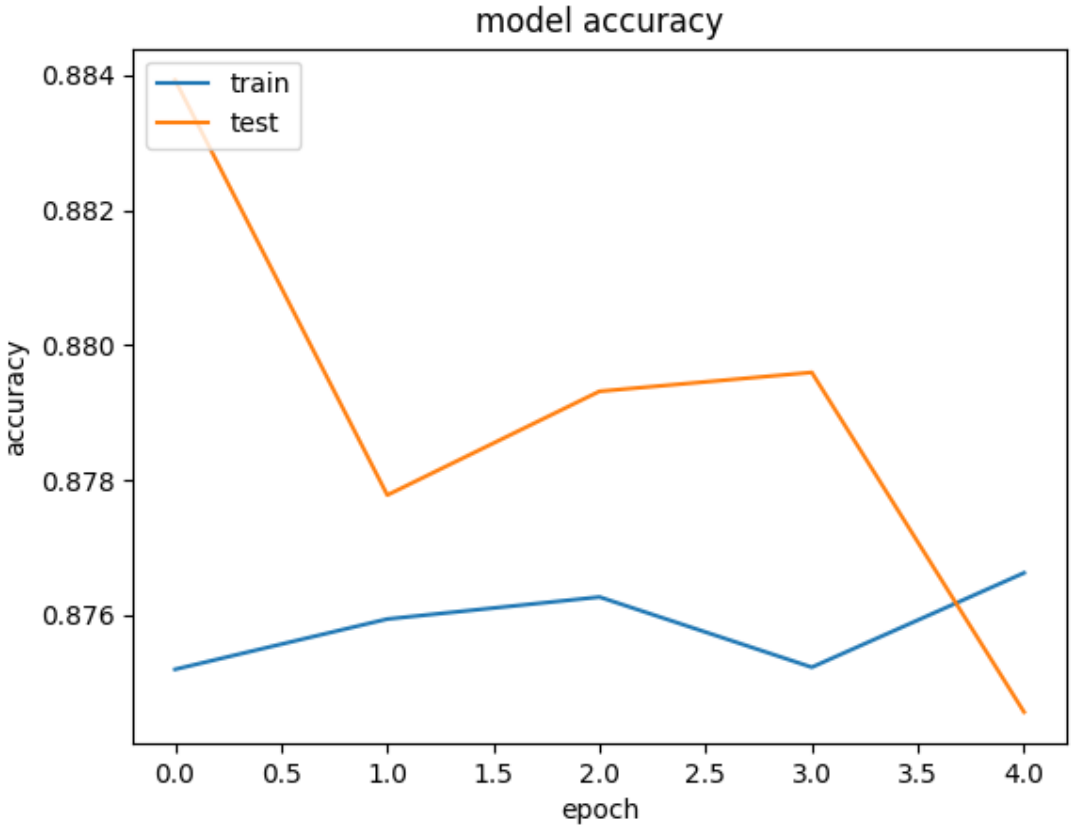

Validation F1 Score Over Folds
Validation F1 Score
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

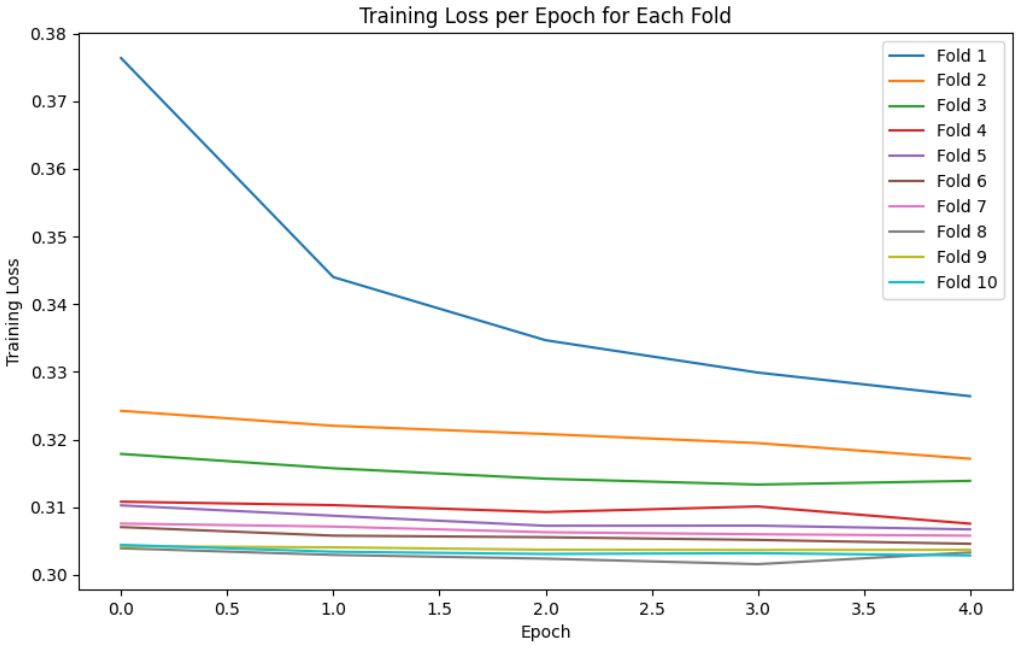

Validation Accuracy Over Folds
Validation Accuracy
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9
10
Folds

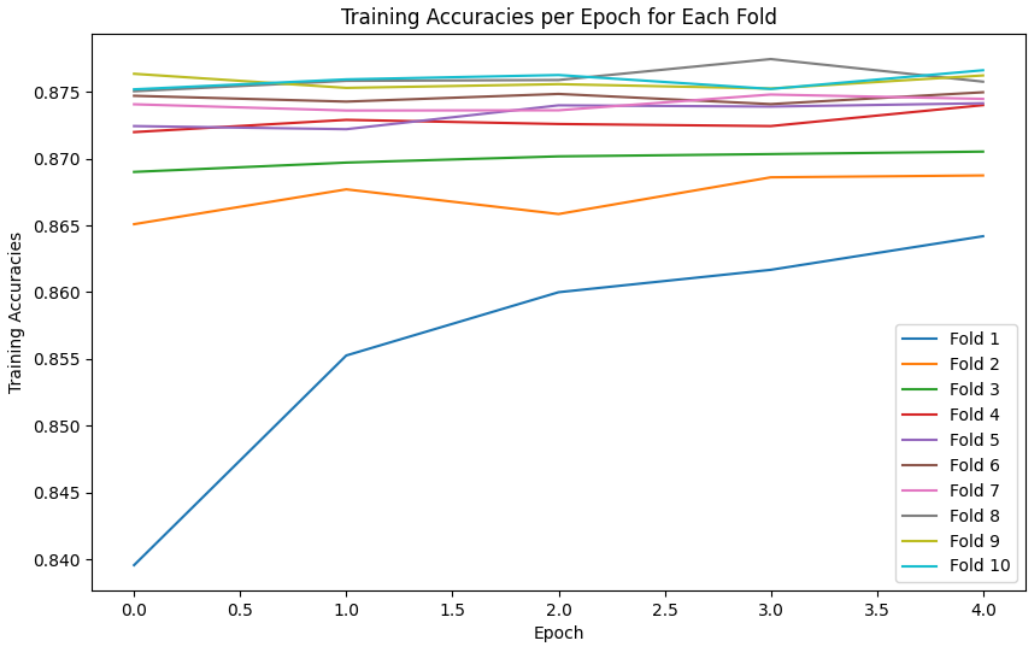
8.5 - Embeddings Estáticos, Modelo GloVe + Fine Tuning:

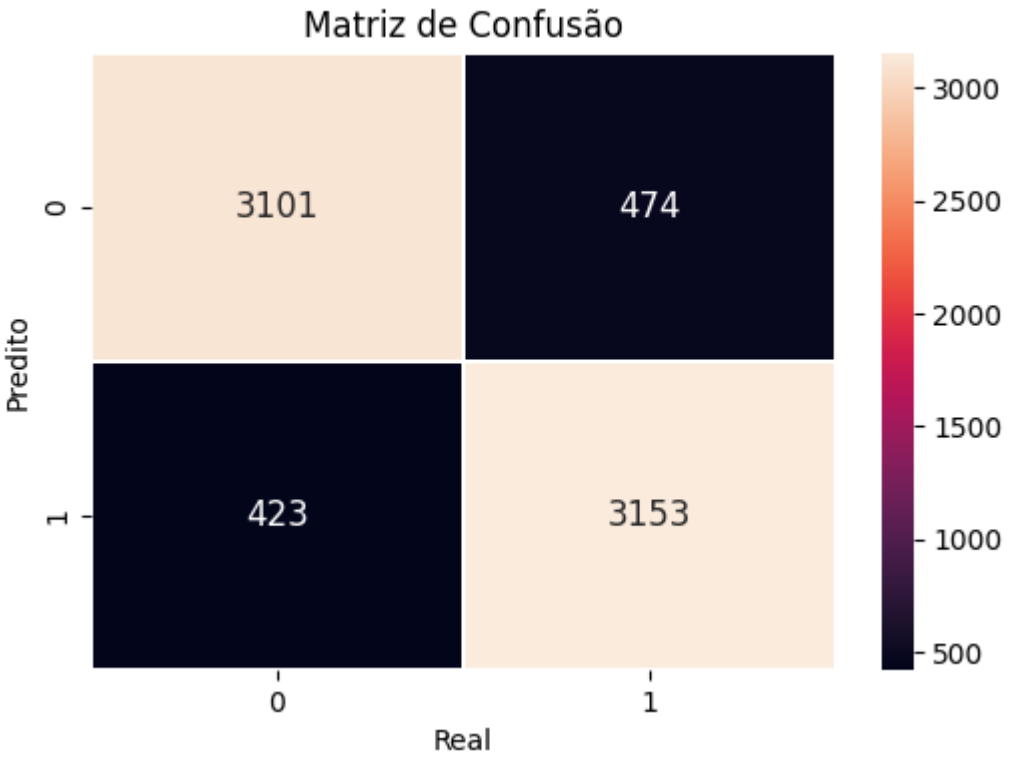
Modelo: GloVe + Fine Tuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.05, Learning rate = 1e-3	0.87 ± 0.01	0.87 ± 0.01	0.87 ± 0.01	0.87 ± 0.01




model accuracy
train
test
accuracy
epoch
0.884
0.882
0.880
0.878
0.876
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Training Loss per Epoch for Each Fold
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Training Loss
Epoch
0.38
0.37
0.36
0.35
0.34
0.33
0.32
0.31
0.30
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0





8.6 - Transformers, Modelo TLM Pequeno + FB1 + MLP:

Modelo: TLM Pequeno + FB1 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0, Learning rate = 5e-5	0.93 ± 0.00	0.93 ± 0.00	0.93 ± 0.00	0.93 ± 0.00

8.7 - Transformers, Modelo TLM Pequeno + FB2 + MLP:

Modelo: TLM Pequeno + FB2 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0, Learning rate = 5e-5	0.93 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.93 ± 0.01

8.8 - Transformers, Modelo TLM Grande + FB1 + MLP:

Modelo: TLM Grande + FB1 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0, Learning rate = 5e-5	1.00 ± 0.01	0.93 ± 0.01	1.00 ± 0.01	0.93 ± 0.01

8.9 - Transformers, Modelo TLM Grande + FB2 + MLP:

Modelo: TLM Grande + FB2 + MLP				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 2.5e-5	0.94 ± 0.00	0.93 ± 0.01	0.94 ± 0.00	0.93 ± 0.01

8.10 - Transformers, Modelo TLM Pequeno + Fine Tuning:

Modelo: TLM Pequeno + FineTuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	0.90 ± 0.00	0.90 ± 0.01	0.90 ± 0.00	0.90 ± 0.01

8.11 - Transformers, Modelo TLM Grande + Fine Tuning:

Modelo: TLM Grande + FineTuning				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	1.00 ± 0.00	0.94 ± 0.00	1.00 ± 0.00	0.94 ± 0.00

8.12 - Transformers, Modelo TLM Pequeno Multilingue:

Modelo: TLM Pequeno Multilíngue				
	Treino		Teste	
Hiperparâmetros	F1 Score	Acurácia	F1 Score	Acurácia
Dropout = 0.10, Learning rate = 5e-5	0.84 ± 0.01	0.83 ± 0.01	0.84 ± 0.01	0.83 ± 0.01

9 - Questões:

9.1 - Modelos com maior quantidade de parâmetros melhoraram a predição? Quando isso não ocorreu? Mostre os resultados.

- No caso citado a seguir, relação de maior quantidade de parâmetros entre modelos e melhor desempenho não se aplicou:

- Modelo FastText + Fine Tuning teve menor quantidade de parâmetros, 7.249, e melhor desempenho na predição com uma acurácia de 0.88 na base buscapé, em relação ao modelo TF-IDF + MLP, que teve 214.585 parâmetros, com acurácia na predição de 0.83 na mesma base.

Na base B2W os dois modelos empataram com 0.93 de acurácia.

Portanto consideramos que o modelo FastText + Fine Tuning com menor quantidade de parâmetros teve melhor desempenho na predição, que o modelo TF-IDF + MLP , com maior quantidade de parâmetros.

Tabela com quantidade de parâmetros: e acurácia na predição:

Modelo	Total de Parâmetros	Acurácia - Base Buscapé	Acurácia - Base B2W
tfidf-mlp	**214.585**	**0.83 ± 0.01**	0.93 ± 0.01
gloVe-mlp	1.075	0.79 ± 0.01	0.86 ± 0.01
gloVe-finetuning	1.075	0.80 ± 0.01	0.87 ± 0.01
fasttext-mlp	7.249	0.86 ± 0.01	0.92 ± 0.01
fasttext-finetuninig	**7.249**	**0.88 ± 0.01**	0.93 ± 0.01
tlm-pequeno-fb1-mlp	58.417	0.86 ± 0.01	0.93 ± 0.00
tlm-pequeno-fb2-mlp	58.417	0.87 ± 0.01	0.93 ± 0.01
tlm-grande-fb1-mlp	24.649	0.87 ± 0.01	0.93 ± 0.01
tlm-grande-fb2-mlp	24.673	0.90 ± 0.02	0.93 ± 0.01
tlm-pequeno-finetuning	18.505	0.90 ± 0.01	0.90 ± 0.01
tlm-grande-finetuning	24.649	0.88 ± 0.00	0.94 ± 0.00
tlm-pequeno-multilingual	18.505	0.84 ± 0.01	0.84 ± 0.01

9.2 - Quais modelos foram os melhores, os piores e quais tiveram empate?

- **Melhores modelos:**

  TLM Grande + Fine Tuning - base B2W

- **Piores modelos:**

- Base Buscapé: GloVe+MLP;
- Base B2W: TLM Multilingue;
- **Modelos empatados:**
    - Base Buscapé: FastText + Fine Tuning e TLM Grande + Fine Tuning
    - Base B2W: TF-IDF-MLP, FastText + Fine Tuning, TLM Pequeno + FB1 + MLP e TLM Pequeno + FB2 + MLP.

9.3 - Ocorreu impacto positivo na predição considerando modelos multilingue em relação aos modelos em português?

- Não ocorreram impactos positivos na predição de modelos multilíngues em relação aos modelos em português. Enquanto modelos multilingue alcançaram a acurácia 0.84, modelos em português superaram com acurácia 0.88.

9.4 - Qual o ranking dos modelos de agregação dos embeddings de Feature Based?

Ranking de modelos FB1 e FB2

Base buscape	Acurácia	Base B2W	Acurácia
TLM Grande + FB2 + MLP	0.90	TLM Pequeno + FB1 + MLP	0.93
TLM Grande + FB1 + MLP	0.87	TLM Pequeno + FB2 + MLP	0.93
TLM Pequeno + FB1 + MLP	0.86	TLM Grande + FB1 + MLP	0.93
TLM Pequeno + FB2 + MLP	0.85	TLM Grande + FB2 + MLP	0.93

Analisando nossos experimentos, constatamos que na base buscapé o modelo TLM Grande+FB2+MLP teve melhor desempenho com acurácia de 0.90 enquanto que na base B2W tivemos empate entre todos os modelos de Feature Based com acurácia de 0.93.

9.5 - Em quais modelos o uso de contexto melhorou a predição e em quais Embeddings estáticos foram competitivos?

- Em treinamentos utilizando o modelo TFIDF+MLP, além de todo o processo de pré-processamento, para tentarmos melhorar o contexto dos corpos das respectivas bases de dados, tentamos ajustar os hiperparâmetros da função TFIDFVectorizer para eliminar frequências de palavras menor que 8 (min_df=8), e também adicionamos o hiperparâmetro max_df=0.3, este hiperparâmetro ignora os termos que tenham uma frequência de documento estritamente superior ao limite determinado. Estas modificações foram realizadas visando tanto a melhora do contexto, a redução da dimensionalidade, e predição;

- Podemos considerar que ocorreram predições competitivas de modelos, na base Buscapé: GloVe+MLP com acurácia de 0.79 e GloVe+Fine Tuning com acurácia de 0.80.

- Podemos considerar que ocorreram predições competitivas de modelos, na base B2W: TF-IDF+MLP com acurácia de 0.93 e FastText+MLP com acurácia de 0.92.

9.6 - Em quais casos o fino ajuste melhorou a predição em relação aos resultados com feature based?

Ranking de modelos FB1 e FB2

Base buscape	Acurácia	Base B2W	Acurácia
TLM Grande + FB2 + MLP	0.90	TLM Pequeno + FB1 + MLP	0.93
TLM Grande + FB1 + MLP	0.87	TLM Pequeno + FB2 + MLP	0.93
TLM Pequeno + FB1 + MLP	0.86	TLM Grande + FB1 + MLP	0.93
TLM Pequeno + FB2 + MLP	0.85	TLM Grande + FB2 + MLP	0.93

Ranking de modelos Fine Tuning

Base buscape	Acurácia	Base B2W	Acurácia
TLM Pequeno + FineTuning	0.90	TLM Grande + FineTuning	0.94
TLM Grande + FineTuning	0.88	Fast Text + Fine Tuning	0.93
Fast Text + Fine Tuning	0.88	TLM Pequeno + FineTuning	0.90
GloVe + Fine Tuning	0.80	GloVe + Fine Tuning	0.87

- Analisando a base buscapé, constatamos que os modelos TLM Grande+FineTuning e FastText+FineTuning melhoraram sua predição com uma acurácia de 0.88 em relação ao modelo TLM Grande+FB1+MLP com acurácia de 0.87;
- Analisando a base B2W, constatamos que o modelo TLM Grande+FineTuning melhorou sua predição com acurácia de 0.94 em relação a todos modelos FB1 e FB2 treinados com a base B2W, que os mesmo obtiveram um empate em 0.93 de acurácia.

9.7 - Compare os resultados do seu trabalho com os do artigo.

Na tentativa de comparar os resultados dos modelos treinados durante os experimentos do projeto e os modelos do artigo, tentamos associar modelos citados no artigo a diferentes modelos abordados no projeto. Por exemplo, tentamos associar o modelo BoW

citado no artigo com o modelo TF-IDF+MLP abordado no projeto. Detalhamos na tabela a seguir as relações e os respectivos valores de métricas de acurácia e f1 score do artigo e do projeto, para fins de comparação:

Comparativos de Resultados do Artigo e Experimentos do Projeto

Resultados do Artigo: Embedding generation for text classification of Brazilian Portuguese user reviews: from bag-of-words to transformers				Resultados de Experimentos do Projeto		
Métrica	**Modelo**	**Buscape**	**B2W**	**Modelo**	**Buscape**	**B2W**
Acurácia	BoW	**0.948 ± 0.02**	0.940 ± 0.03	TF-IDF+MLP	0.83 ± 0.01	**0.93 ± 0.01**
	CNN	**0.957 ± 0.02**	0.947 ± 0.06	GloVe+MLP	0.79 ± 0.01	0.86 ± 0.01
	LSTM	**0.955 ± 0.02**	0.944 ± 0.01	GloVe+Fine Tuning	0.80 ± 0.01	0.87 ± 0.01
	CNN	**0.957 ± 0.02**	0.947 ± 0.06	FastText+MLP	0.86 ± 0.01	0.92 ± 0.01
	LSTM	**0.955 ± 0.02**	0.944 ± 0.01	FastText+Fine Tuning	0.88 ± 0.01	**0.93 ± 0.01**
	FB TLM	**0.948 ± 0.02**	0.940 ± 0.03	TLM Peq. + FB1 + MLP	0.86 ± 0.01	**0.93 ± 0.00**
	FB TLM	**0.948 ± 0.02**	0.940 ± 0.03	TLM Peq. + FB2 + MLP	0.87 ± 0.01	**0.93 ± 0.01**
	FT TLM	**0.978 ± 0.01**	0.978 ± 0.01	TLM Peq. + Fine Tuning	0.90 ± 0.01	0.90 ± 0.01
	FT TLM	**0.978 ± 0.01**	0.978 ± 0.01	TLM Grande + Fine Tuning	0.88 ± 0.00	**0.94 ± 0.00**
	FT TLM	**0.978 ± 0.01**	0.978 ± 0.01	TLM Multilingue	0.84 ± 0.01	0.84 ± 0.01
F1-Score	BoW	**0.972 ± 0.01**	0.957 ± 0.02	TF-IDF+MLP	0.83 ± 0.01	**0.93 ± 0.01**
	CNN	**0.977 ± 0.01**	0.962 ± 0.04	GloVe+MLP	0.79 ± 0.01	0.87 ± 0.01
	LSTM	**0.976 ± 0.01**	0.960 ± 0.06	GloVe+Fine Tuning	0.80 ± 0.01	0.87 ± 0.01
	CNN	**0.977 ± 0.01**	0.962 ± 0.04	FastText+MLP	0.87 ± 0.01	0.92 ± 0.01
	LSTM	**0.976 ± 0.01**	0.960 ± 0.06	FastText+Fine Tuning	0.88 ± 0.01	**0.93 ± 0.01**
	FB TLM	**0.976 ± 0.01**	0.972 ± 0.01	TLM Peq. + FB1 + MLP	0.86 ± 0.01	**0.93 ± 0.00**
	FB TLM	**0.976 ± 0.01**	0.972 ± 0.01	TLM Peq. + FB2 + MLP	0.87 ± 0.01	**0.93 ± 0.01**
	FT TLM	**0.978 ± 0.01**	0.978 ± 0.01	TLM Peq. + Fine Tuning	0.90 ± 0.00	0.90 ± 0.00
	FT TLM	0.978 ± 0.01	0.978 ± 0.01	TLM Grande + Fine Tuning	0.89 ± 0.00	**1.00 ± 0.00**
	FT TLM	0.978 ± 0.01	0.978 ± 0.01	TLM Multilingue	0.84 ± 0.01	0.84 ± 0.01

Analisando as métricas de acurácia e f1 score do artigo, dos modelos BoW, CNN, LSTM, FB TLM, FT TML, constatamos que tiveram um ótimo desempenho em relação aos modelos TFIDF+MLP, GloVe+MLP, GloVe+Fine Tuning, FastText+MLP, FastText+Fine Tuning, e as variações de Transformers que foram comparadas.

O modelo BoW citado no artigo alcançou acurácia de 0.94 na base buscapé, enquanto o modelo TF-IDF+MLP treinado no projeto alcançou acurácia de 0.93 na base b2w.

O modelo FT TLM citado no artigo teve o melhor desempenho na base buscapé e b2w com 0.97 de acurácia.

O modelo TLM Grande + Fine Tuning teve o melhor desempenho na base B2w com 0.94 de acurácia, e o modelo TLM pequeno + Fine Tuning teve o melhor desempenho na base Buscapé.

Analisando valores de métricas dos experimentos citados no artigo, constatamos que todos os modelos tiveram um excelente desempenho, todos com acurácia e F1 score acima de 0.90. Porém analisando os valores de métricas de experimentos do projeto, constatamos que tiveram alguns modelos que tiveram um desempenho regular, por exemplo, Glove+MLP, GloVe+FineTuning, TLM Multilingue, variando de 0.84 a 0.86 de acurácia.

9.8 - Como a representação pré-treinada foi construída? Descreva os métodos de treinamento, base de dados utilizada e dimensões do modelo transformer.

Os modelos de representações pré-treinadas designados para nosso grupo foram:

- **XML-Roberta-Base:**
- Modelo RoBERTa (A Robustly Optimized BERT Pre Training Approach), é constituído utilizando a arquitetura Transformer, que é um tipo de rede neural projetada para processar sequências de dados, como por exemplo texto;
- Ela é baseada no Google's BERT;
- Release do modelo publicada no ano 2018;
- Algumas características são:
    - Remoção do objetivo de pré-treinamento da próxima frase;
    - Treinamento com mini lotes e taxas de aprendizagem muito maiores;
    - Tamanho do vocabulário:

```
# Importando xml-roberta-base
tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-base")
model = AutoModel.from_pretrained("xlm-roberta-base")
```

```
print(f'Quantidade de tokens que xml-roberta-base tem no vocabulário: {len(tokenizer.vocab)}')
```

```
Quantidade de tokens que xml-roberta-base tem no vocabulário: 250002
```

    - Tipos de tokens:

```
tokenizer.special_tokens_map
```

```
{'bos_token': '<s>',
 'eos_token': '</s>',
 'unk_token': '<unk>',
 'sep_token': '</s>',
 'pad_token': '<pad>',
 'cls_token': '<s>',
 'mask_token': '<mask>'}
```

```
print('bos_token_id <s>:', tokenizer.bos_token_id)
print('eos_token_id </s>:', tokenizer.eos_token_id)
print('sep_token_id </s>:', tokenizer.sep_token_id)
print('pad_token_id <pad>:', tokenizer.pad_token_id)
```

```
bos_token_id <s>: 0
eos_token_id </s>: 2
sep_token_id </s>: 2
pad_token_id <pad>: 1
```

- 768 dimensões (obs.: valor inicial é número de instâncias da base de dados Buscapé)

```
(133632, 768)
(133632, 768)
(133632, 768)
```

-

- **XML-Roberta-Large**
- Large-sized-model;
- Pré-treinado em 2,5TB de dados CommonCrawl, filtrados contendo 100 idiomas;
- Pré-treinado apenas nos textos brutos, sem nenhum tipo de rótulo humano, é por isso que ele pode usar muitos dados disponíveis publicamente;
- Contém processo automático para gerar entradas e rótulos a partir desses textos;

```
print(f'Quantidade de tokens que xml-roberta-large tem no vocabulário: {len(tokenizer.vocab)}')

Quantidade de tokens que xml-roberta-large tem no vocabulário: 250002
```

```
[6] tokenizer.special_tokens_map

{'bos_token': '<s>',
 'eos_token': '</s>',
 'unk_token': '<unk>',
 'sep_token': '</s>',
 'pad_token': '<pad>',
 'cls_token': '<s>',
 'mask_token': '<mask>'}

print('bos_token_id <s>:', tokenizer.bos_token_id)
print('eos_token_id </s>:', tokenizer.eos_token_id)
print('sep_token_id </s>:', tokenizer.sep_token_id)
print('pad_token_id <pad>:', tokenizer.pad_token_id)

bos_token_id <s>: 0
eos_token_id </s>: 2
sep_token_id </s>: 2
pad_token_id <pad>: 1
```

- 1024 dimensões;

- **BERT-Base**
- Substituímos o BERT-Large multilingue pelo Bert base multilingual cased, pelo fato de não encontramos Bert Large Multilingual;
- Modelo pré-treinado nos 104 principais idiomas;
- Masked Language Modeling (MLM): pegando uma frase, o modelo máscara aleatoriamente 15% das palavras na entrada, em seguida executa toda a frase mascarada no modelo, e tenta prever as palavras mascaradas;
- Diferencia letras maiúsculas de minúsculas;
  - Segue algumas configurações do BERT Base Multilingual:
  - 768 dimensões;
  - Quantidade do vocabulário:

```
[5] print(f'Quantidade de tokens que bert-base-multilingual-cased tem no vocabulário: {len(tokenizer.vocab)}')

    Quantidade de tokens que bert-base-multilingual-cased tem no vocabulário: 119547

    tokenizer.special_tokens_map
```

  - Detalhamento dos tokens:

```
[6] tokenizer.special_tokens_map

    {'unk_token': '[UNK]',
     'sep_token': '[SEP]',
     'pad_token': '[PAD]',
     'cls_token': '[CLS]',
     'mask_token': '[MASK]'}

    print('unk_token_id <s>:', tokenizer.unk_token_id)
    print('sep_token_id </s>:', tokenizer.sep_token_id)
    print('pad_token_id </s>:', tokenizer.pad_token_id)
    print('cls_token_id <pad>:', tokenizer.cls_token_id)
    print('mask_token_id <pad>:', tokenizer.mask_token_id)

    unk_token_id <s>: 100
    sep_token_id </s>: 102
    pad_token_id </s>: 0
    cls_token_id <pad>: 101
    mask_token_id <pad>: 103
```