Projeto Text Similarity

António Augusto Fernandes Simões Pereira

Nº 21136 – Regime Pós-laboral

Orientação

Prof. Joaquim Gonçalves

Prof. Patrícia Leite

Ano letivo 2022/2023

Licenciatura em Engenharia de Sistemas Informáticos

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e do Ave

Identificação do Aluno

António Augusto Fernandes Simões Pereira

Aluno número 21136, regime pós-laboral

Licenciatura em Engenharia de Sistemas Informáticos

Orientação

Prof. Joaquim Gonçalves

Prof. Patrícia Leite

RESUMO

As empresas vão investindo na automatização de processos internos para a redução de custos, surgindo a possibilidade de poder fazer uma análise de reclamações de uma empresa através da similaridade de texto e, assim, automatizar este processo. Isto foi possível através da realização de cálculos de semelhança entre frases utilizando as variadas ferramentas que foram estudadas.

Foram então tiradas conclusões dos testes realizados, encaminhando ao cumprimento dos objetivos inicialmente estabelecidos. Os resultados não só demonstram a viabilidade da utilização da semelhança de texto para a análise de reclamações, como também salientam o êxito da realização dos objetivos do projeto. Esta investigação contribui para os esforços em curso no sentido de tirar partido da automatização para uma gestão eficaz dos processos internos das empresas.

ABSTRACT

Companies are increasingly investing in the automation of internal processes to reduce costs, giving rise to the focus of our project: addressing the opportunity to analyze company complaints through text similarity and automate this process.

Various tools were used to calculate similarity between sentences, store and present information to the user, and ultimately conduct tests and analyze the results. Through these tools, we aimed to develop an efficient system for automating the analysis of complaints.

Following a thorough analysis of the tests, conclusive findings were drawn, aligning with the initially established objectives. The outcomes not only demonstrate the viability of utilizing text similarity for complaint analysis but also underscore the successful achievement of the project's goals. This research contributes to the ongoing efforts in leveraging automation for effective internal process management within companies.

ÍNDICE

1. Introdução X

1.1. Objetivos 1

1.2. Contexto 1

1.3. Estrutura do documento 1

2. Estado da arte 3

2.1. Re-LSTM: A long short-term memory network text similarity algorithm based on weighted word embedding. (Weidong Z., 2022) 3

2.2. A Short Text Similarity Calculation Method Combining Semantic and Headword Attention Mechanism (Mingyu J., 2022) 4

2.3. Computing semantic similarity of texts based on deep graph learning with ability to use semantic role label information. (Majid M., 2022) 5

2.4. A fast text similarity measure for large document collections using multireference cosine and genetic algorithm. (Mohammadi, 2020) 6

3. Tecnologias utilizadas 7

3.1. Sklearn 7

3.2. Python 7

3.3. Flask 7

3.4. MongoDB 8

3.5. Kotlin 8

4. Trabalho Desenvolvido 9

4.1. Backend 9

4.2. Mobile App 14

4.3. Testes e Resultados 17

4.3.1. Primeiro teste 18

4.3.1.1. Categoria Esperada - Facility Related 18

4.3.1.2. Categoria Esperada - Product Related 19

4.3.1.3. Categoria Esperada - Other 20

4.3.2. Segundo Teste 20
4.3.2.1. Categoria Esperada – Facility Related 20
4.3.2.2. Categoria Esperada – Product Related 21
4.3.2.3. Categoria Esperada - Other 21
4.4. Conclusões 22
5. Conclusão 23
6. Bibliografia XXIV

ÍNDICE DE FIGURAS

Figura 1 – Código relativo a categories……9

Figura 2 – Exemplo de reclamações de treino……10

Figura 3 – Exemplo de categorias de treino……11

Figura 4 – Criação do modelo……...11

Figura 5 – Método GET da API Flask para obter reclamações……12

Figura 6 – Função que lê o ficheiro de texto com as reclamações de teste……12

Figura 7 – Ligação à base de dados MongoDB……..12

Figura 8 – Funções de cálculo de similaridade e armazenamento de informação……..13

Figura 9 – Mockup da Mobile App……14

Figura 10 – Função que utiliza a Flask API para receber a informação armazenada na base de dados……15

Figura 11 – Função que altera na view a informação recebida da base de dados……...16

Siglas e Acrónimos

NLP – Natural Language Processing (Processamento de linguagem natural)

DDS – Drug-Drug Similarity

LSTM – Long Short-Term Memory

HA-RCNN - Hierarchical Attentive Recurrent Convolutional Neural Network

DGNN – Directed Graph Neural Network

SRL - Semantic Role Labelling

TF-IDF - Term Frequency-Inverse Document Frequency

1. Introdução

O processamento de linguagem natural (NLP) é uma das áreas da inteligência artificial que visa a interpretação e produção de linguagem natural (fala e escrita humana), por parte de computadores. Uma das funções do NLP é a similaridade de texto que envolve medir o grau de semelhança entre dois textos. (Lemos, Figueiredo, & Botler, 2022) A similaridade de texto é extremamente importante para aplicações como verificação de plágio ou identificação de documentos duplicados, mas também pode ser aplicado a programas de reclamações ou de informações de medicamentos. (Lemos, Figueiredo, & Botler, 2022)

Neste projeto foram exploradas as bases da similaridade de texto, e as suas principais técnicas e ferramentas utilizadas. Foram estudados diferentes algoritmos de processamento de linguagem natural e como estes podem ser utilizados para medir a similaridade entre textos de variados temas e explicação das abordagens para a avaliação desses algoritmos.

1.1. Objetivos

Os objetivos deste projeto é a criação de uma aplicação capaz de reconhecer palavras-chave através de métodos que estão introduzidos na área da similaridade de texto e apresentar a informação recolhida num dispositivo móvel e também a assimilação dos vários conceitos da área da similaridade de texto.

1.2. Contexto

Este projeto final de licenciatura orientado pelo professor Joaquim Gonçalves e pela professora Patrícia Leite originou da necessidade de automatizar um processo em que seja feito um cálculo de similaridade entre dois textos e tem como finalidade aprender novos conceitos da área em questão.

1.3. Estrutura do documento

Em relação à estrutura do documento estes são os principais tópicos;

1. Introdução – onde é feita a introdução ao projeto e onde são apresentados os objetivos e contexto do mesmo;
2. Estado da arte – pequena introdução ao estado da arte do tema do projeto e análise de vários artigos;
3. Implementação – onde se descrevem as tecnologias escolhidas (e se justifiquem).
4. Trabalho Desenvolvido - análise do código desenvolvido, explicação dos testes e análise dos resultados.
5. Conclusão.

2. Estado da arte

Com a crescente procura de aplicações que utilizem a similaridade de texto nos diversos setores, o investimento na pesquisa da mesma é extremamente importante para que novas tecnologias, que são necessárias no nosso dia a dia, sejam desenvolvidas e aprimoradas. (Nora, Moncy, P, Sreedarsana, & Sindhya)

No contexto de *text similarity* são utilizados diferentes modelos de redes neurais, que são redes artificiais inspiradas no cérebro humano e têm vários usos como o reconhecimento de padrões e o processamento de linguagem natural. Estas redes neurais podem possuir mecanismos de atenção que permitem ao modelo focar-se em partes específicas dos dados de entrada atribuindo pesos diferentes a palavras diferentes do texto dado. (Viji & Revathy, 2022)

Todos os algoritmos explorados têm um nível de complexidade diferente, mas todos têm como base a similaridade de texto e podem ser melhores em situações diferentes. A vantagem do uso deste tipo de algoritmos está na facilidade de uso dos mesmos.

2.1. Re-LSTM: A long short-term memory network text similarity algorithm based on weighted word embedding. (Weidong Z., 2022)

Os autores deste artigo utilizaram o algoritmo Re-LSTM (*Long Short-Term Memory*) como o seu modelo de computação de similaridade de texto, que utiliza uma rede neural recorrente (LSTM) para extrair características implícitas no texto. Este modelo utiliza uma abordagem de incorporação palavras ponderada através do algoritmo χ⊃2-C que avalia as diferentes palavras encontradas no texto através da sua categoria e frequência com que aparece no documento.

Outro método utilizado neste modelo é o TF-IDF (*Term Frequency-Inverse Document Frequency*) que é uma técnica utilizada no processamento de linguagem natural para medir a importância relativa de uma palavra presente em um ou vários documentos. Isto é possível calculando o peso de cada palavra com base na quantidade de vezes que aparece num documento específico ou vários documentos do mesmo tema. Se uma palavra aparece várias vezes num documento vai ter um peso maior pois é mais importante nesse documento específico.

2.2. A Short Text Similarity Calculation Method Combining Semantic and Headword Attention Mechanism (Mingyu J., 2022)

Segundo os autores, este modelo teve um comportamento muito bom em relação a outros modelos com mecanismos de atenção, ainda que a dimensão da amostra utilizada no *dat*aset[1] seja reduzida. o que os leva a acreditar que com uma amostra maior os resultados poderiam ser melhores também (ATEC utilizado para detetar fraudes discais e MSRP com o preço de carros indicado pelo fabricante e usado para tentar determinar o seu preço no futuro).

O modelo de rede neural usado é o *Hierarchical Attentive Recurrent Convolutional Neural Network* (HA-RCNN) que consiste na hierarquização das palavras em árvore através do seu "mecanismo de atenção" que se concentra nas partes importantes de um texto.

[1] Um conjunto de dados para treino do modelo

2.3. Computing semantic similarity of texts based on deep graph learning with ability to use semantic role label information. (Majid M., 2022)

O modelo de rede neural utilizado pelos autores neste projeto é o *Directed Graph Neural Network* (DGNN) que trabalha com grafos de relação semântica (*Semantic Role Labelling* – SRL) para calcular a similaridade semântica do texto (*Text Similarity*).

Neste caso os autores fizeram dois testes um com SRL + DG (um grafo direcional convencional) e outro com SRL + SDG (um grafo em que todos os tipos de arestas são considerados como um único tipo). Os resultados observados permitiram aos autores concluir que o segundo tipo de grafo aumenta o desempenho do algoritmo em relação a um grafo convencional devido.

2.4. A fast text similarity measure for large document collections using multireference cosine and genetic algorithm. (Mohammadi, 2020)

Os autores deste artigo falam-nos sobre a importância da utilização da similaridade de texto para o aumento do desempenho dos motores de busca como por exemplo o Google.

Para que o desempenho de um motor de busca aumente é preciso remover todas as pesquisas com artigos/textos que sejam iguais ou similares para que diminua o número de resultados presentes no *index* resultando num tempo de pesquisa menor e menor probabilidade de encontrar informação repetida.

Os métodos utilizados neste tipo de *text similarity* são chamados de *Duplicate or Near-Duplicate* – DND que, tal como o nome indica, são utilizados para encontrar artigos/textos idênticos para que não seja apresentada informação duplicada. Neste projeto foi utilizado o *cosine text similarity algorithm* que cria, através da análise de várias partes de um artigo, um vetor *term frequency-inverse document frequency* (TF-IDF. Através deste vetor é possível saber se um outro texto que for analisado é igual ou parecido se o angulo dos vetores for parecido.

3. Tecnologias utilizadas

A escolha das tecnologias utilizadas é crucial para atingir os objetivos estabelecidos no projeto, pois cada uma destas tecnologias desempenha um papel fundamental na construção e operação do mesmo. Uma vez que a experiência com este tipo de aplicações era pouca as tecnologias foram escolhidas com base na facilidade de uso das mesmas.

3.1. Sklearn

É uma biblioteca python que tem à disposição vários algoritmos de machine learning, como é o caso do método de Naive Bayes e por não ter experiência com este tipo de aplicações optei pelo uso desta biblioteca pela sua facilidade de uso.

É necessária a utilização da biblioteca Sklearn para que seja calculado o grau de similaridade das várias frases dadas utilizando o modelo treinado e, assim, seja atribuída a cada frase a sua respetiva categoria.

Esta biblioteca foi utilizada no *backend* do projeto e foi responsável pelos cálculos da similaridade entre frases.

3.2. Python

A linguagem de programação escolhida foi o Python devido à maior disponibilidade de documentção sobre o tema do projeto que trata a similaridade de texto (NLP) em relação a outras linguagens de programação.

3.3. Flask

Para que a aplicação backend em python que trata do cálculo da similaridade de texto consiga comunicar e enviar informação para o frontend foi necessário escolher uma api capaz de atender as necessidades do projeto e, por isso foi escolhida a biblioteca Flask por ser eficiente, simples e fácil de integrar no projeto.

3.4. MongoDB

A base de dados escolhida para este trabalho foi a mongodb visto que a informação que estamos a guardar tem apenas dois atributos, a frase da reclamação a ser analisada e a respetiva categoria atribuída pelo modelo. A exportação da informação é feita através de um ficheiro json o que facilita a sua utilização na aplicação móvel.

Esta ferramenta foi utilizada para receber informação do *backend* e posteriormente enviá-la para a aplicação móvel em Kotlin.

3.5. Kotlin

Para a criação da aplicação móvel foi utilizada a linguagem Kotlin uma vez que é a linguagem que foi abordada na cadeira de Programação de Dispositivos Móveis e, por isso, utilizei os conhecimentos adquiridos na mesma para produzir uma aplicação móvel que permite a visualização da informação dada pela aplicação original.

4. Trabalho Desenvolvido

O trabalho desenvolvido começa pela construção do *backend* e *frontend* onde é explicado o seu código e forma como é estruturado e depois, é feita a explicação dos testes realizados e análise de resultados.

4.1. Backend

Em relação ao *backend* foi feito um modelo de treino que pode conter várias categorias e era treinado dando uma frase e a sua respetiva categoria. Depois do modelo estar treinado é analisada cada frase de um ficheiro de texto externo e atribuída a respetiva categoria para cada uma das frases.

```
categories = [
    "facility_related",
    "product_related",
    "other",
]
```

Figura 1 – Código relativo a categories

Na Figura 1 acima apresentada podemos ver um excerto de código e que estão representadas as possíveis categorias que podem ser atribuídas às frases (reclamações sobre as instalações, reclamações sobre o produto e reclamações que não pertençam às duas primeiras categorias).

```
    training_complaints = [
        "The stairs are too steep and dangerous.",
        "The restroom is not clean and needs maintenance.",
        "The elevator is out of order.",
        "The parking lot is always full.",
        "The chairs in the waiting area are uncomfortable.",
        "The lighting in the hallways is too dim.",
        "Some other complaint not related to specific facilities.",
        "Another generic complaint.",
    ]
```

Figura 2 – Exemplo de reclamações de treino

Na Figura 2 temos representado um exemplo das reclamações de treino que serão utilizadas para treinar o modelo. As frases aqui apresentadas são um exemplo pelo que podem ser utilizadas frases diferentes conforme o teste a ser realizado.

```
training_categories = [
    "facility_related",
    "facility_related",
    "facility_related",
    "facility_related",
    "facility_related",
    "facility_related",
    "other",
    "other",
]
```

Figura 3 – Exemplo de categorias de treino

E, por fim, na figura 3 temos um exemplo das categorias que correspondem a cada frase de treino utilizada acima.

```
model = make_pipeline(CountVectorizer(), MultinomialNB())
```

Figura 4 – Criação do modelo

Na Figura 4 está representada a criação do modelo utilizado na aplicação em que são utilizados dois métodos da biblioteca sklearn que são o Count Vectorizer que simplesmente transforma cada palavra numa frase dada em um token e conta as vezes que o mesmo token aparece e o método MultinomialNB utiliza o teorema de Naive Bayes e a informação dada pelo Count Vectorizer para atribuir uma categoria.

```
@app.route('/complaints', methods=['GET'])
def get_complaints():
    complaints_data = list(collection.find({}, {'_id': 0}))
# Retrieve all complaints data
    return jsonify(complaints_data), 200
```

Figura 5 – Método GET da API Flask para obter reclamações

A api criada pelo flask tem apenas um método GET, que está representado na Figura 5, para obter todos os *complaints* que foram analisados e guardados na base de dados e será utilizada pela aplicação mobile para esta receber a informação.

```
with open('test_complaints.txt', 'r') as file:
        test_complaints = file.read().splitlines()
```

Figura 6 – Função que lê o ficheiro de texto com as reclamações de teste

Na Figura 6 está representada uma função que lê o ficheiro em que estão as frases que vão ser testadas e trata da divisão das frases por linhas.

```
client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["projectdb"]
collection = db["complaints"]
```

Figura 7 – Ligação à base de dados MongoDB

Na Figura 7 estão representadas as variáveis que realizam a conexão à base de dados para que mais tarde seja possível armazenar informação na mesma.

```
        for idx, complaint in enumerate(test_complaints, 1):
            category = predict_category(complaint)
            print(f"Complaint  {idx}:  {complaint}\nCategory:
{category}\n")
            store_in_mongodb(complaint, category)


    def predict_category(complaint):
        predicted_category = model.predict([complaint])
        return predicted_category[0]


    def store_in_mongodb(complaint, category):
        complaint_data = {
            "complaint": complaint,
            "category": category
        }
        collection.insert_one(complaint_data)
```

Figura 8 – Funções de cálculo de similaridade e armazenamento de informação

Na figura 8 é utilizado um *for* que itera por todas as frases e em cada uma delas atribui uma categoria utilizando a função *predict_category* e, de seguida armazena na base de dados com a função *store_in_mongodb*.

4.2. Mobile App

A mobile app foi utilizada apenas para mostrar informação que é recebida através da flask API do *backend*. O layout da aplicação contém apenas uma lista onde são adicionados itens que são recebidos da api.

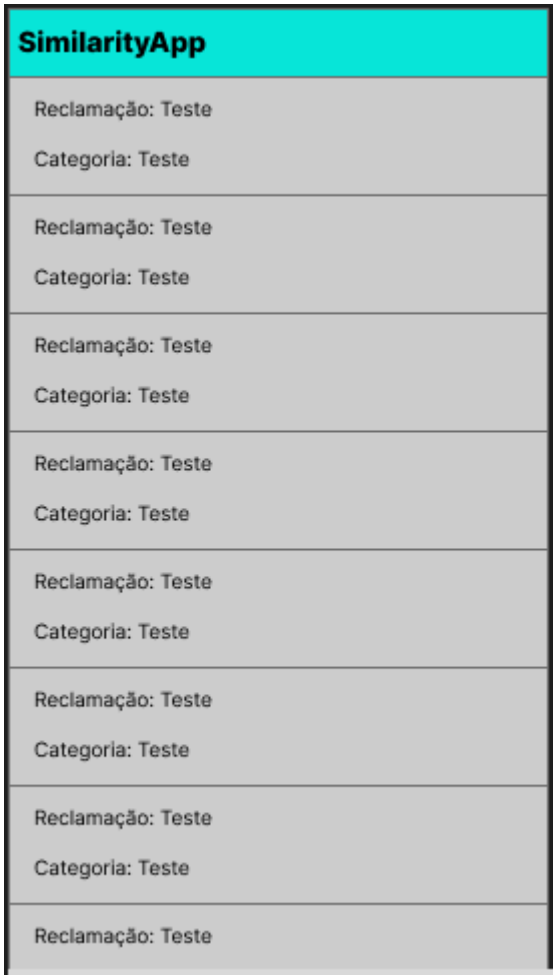

Figura 9 – Mockup da Mobile App

```
    fun fetchDataFromApi() {
            GlobalScope.launch(Dispatchers.IO) {
                try {
                    val apiResponse = HttpUtils.fetchDataFromApi("localhost:5000/complaints")
                    val complaintsList = parseJsonResponse(apiResponse)
                    _complaints.postValue(complaintsList)
                } catch (e: Exception) {

                }
            }
        }
```

Figura 10 – Função que utiliza a Flask API para receber a informação armazenada na base de dados

Na Figura 10 está representado um excerto de código que faz um pedido à API para receber as informações anteriormente guardadas na base de dados.

```
    override fun getView(position: Int, convertView: View?,
parent: ViewGroup?): View {
            var convertView = convertView
            if (convertView == null) {
                val                 inflater                =
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE)    as
LayoutInflater
                convertView                                 =
inflater.inflate(R.layout.list_item, null)
            }

            val        fraseTextView:        TextView        =
convertView!!.findViewById(R.id.fraseTextView)
            val       categoriaTextView:       TextView       =
convertView.findViewById(R.id.categoriaTextView)

            val data = dataList[position]

            fraseTextView.text = data.frase
            categoriaTextView.text = data.categoria

            return convertView
        }
```

Figura 11 – Função que altera na view a informação recebida da base de dados

Na Figura 11 está representada uma função que utiliza a informação recebida da base de dados para apresentar os dados na aplicação móvel.

4.3. Testes e Resultados

Para vermos a eficácia do modelo apresentado foram realizados vários testes variando a quantidade de frases de treino. As frases que vão ser utilizadas para treino são as seguintes:

Facility Related

The air conditioning in the waiting area is consistently too cold, making it uncomfortable for visitors.

The Wi-Fi connection in the facility is unreliable, causing inconvenience for those who rely on it.

The signage in the building is unclear, leading to confusion for visitors trying to navigate the facility.

The conference rooms lack proper audio-visual equipment, hindering effective presentations.

The carpeting in the hallways is worn out and needs replacement to enhance the overall appearance.

Product Related

The new software update has caused frequent crashes and disruptions in our workflow.

The product warranty is not clearly explained, causing confusion among customers regarding coverage.

The product manuals are poorly written, making it challenging for users to understand its features.

The product assembly instructions are incomplete, causing frustration for customers attempting to set it up.

The customer support for the product is unresponsive, leaving users without timely assistance.

Other

The website's customer service chatbot is not providing helpful responses.

The response time for resolving customer inquiries is excessively long.

The company's social media presence lacks engagement and responsiveness.

The company's return policy is unclear and needs better communication to customers.

The product delivery times are inconsistent and often delayed.

4.3.1. Primeiro teste

Para o primeiro teste foi utilizada apenas a primeira frase de treino de cada uma das categorias apresentadas acima no modelo de treino, por ser a quantidade mínima a ser testada para cada uma das três frases de teste de cada categoria apresentadas abaixo:

4.3.1.1. Categoria Esperada - Facility Related

Complaint 1: The restroom facilities are outdated and in need of modernization.

Category: product_related

Complaint 2: The hand sanitizer dispensers are frequently empty, posing a hygiene concern in the facility.

Category: facility_related

Complaint 3: There is a persistent leak in the ceiling near the entrance, creating a safety hazard.

Category: facility_related

As primeiras três reclamações deveriam ser *facility related*, mas como há apenas uma frase para teste em cada uma das categorias o modelo de treino não tem informação suficiente para responder corretamente a categoria.

4.3.1.2. Categoria Esperada - Product Related

Complaint 4: The latest product release has a significant decrease in quality compared to previous versions.

Category: product_related

Complaint 5: The packaging of the product is flimsy, leading to damage during shipping.

Category: facility_related

Complaint 6: The user interface of the application is confusing, making it difficult for customers to navigate.

Category: facility_related

No caso das *complaints* 4,5 e 6 que deveriam ser todas as frases pertencentes à categoria *product related* também houve uma diferença na categoria atribuída pelo modelo devido também à falta de frases de treino.

4.3.1.3. Categoria Esperada - Other

Complaint 7: The company's promotional emails are too frequent and annoying.

Category: product_related

Complaint 8: The website frequently experiences downtime, affecting online services.

Category: other

Complaint 9: The company's billing system is confusing, leading to overcharges for some customers.

Category: facility_related

Nas 3 últimas reclamações apenas a reclamação 8 obteve a categoria correta e mais uma vez pelo baixo número de frases no modelo de treino.

4.3.2. Segundo Teste

No segundo teste adicionei as restantes frases de treino de cada categoria ao modelo e testei nas mesmas frases de teste para ver se os resultados variavam:

4.3.2.1. Categoria Esperada – Facility Related

Complaint 1: The restroom facilities are outdated and in need of modernization.

Category: facility_related

Complaint 2: The hand sanitizer dispensers are frequently empty, posing a hygiene concern in the facility.

Category: facility_related

Complaint 3: There is a persistent leak in the ceiling near the entrance, creating a safety hazard.

Category: facility_related

Após a adição de mais frases de treino as categorias das reclamações corresponderam ao esperado e a informação devolvida foi a correta, o que já era esperado devido a uma maior amostra de frases a serem analisadas para treino.

4.3.2.2. Categoria Esperada – Product Related

Complaint 4: The latest product release has a significant decrease in quality compared to previous versions.

Category: product_related

Complaint 5: The packaging of the product is flimsy, leading to damage during shipping.

Category: product_related

Complaint 6: The user interface of the application is confusing, making it difficult for customers to navigate.

Category: other

Para a categoria product related apenas uma das frases estava errada significando que o algoritmo melhorou, mas seria necessária uma maior amostra para que fosse 100% eficaz.

4.3.2.3. Categoria Esperada - Other

Complaint 7: The company's promotional emails are too frequent and annoying.

Category: other

Complaint 8: The website frequently experiences downtime, affecting online services.

Category: other

Complaint 9: The company's billing system is confusing, leading to overcharges for some customers.

Category: product_related

Como na categoria anterior o algoritmo falhou apenas em uma das frases testadas sendo mais eficaz que o primeiro teste, mas ainda não prevê todas as categorias de forma acertada.

4.4. Conclusões

Depois da realização dos testes podemos concluir que o algoritmo poderá ser utilizado numa situação real pressupondo que o nível de amostras que são dadas ao modelo de testes for grande e, assim, garantir que irá atribuir de forma correta a categoria de cada reclamação que é recebida de cada utilizador dos serviços da empresa em questão.

A necessidade de uma grande amostra pode ser uma desvantagem para uma empresa que quer começar a utilizar o algoritmo e não tem dados que alimentem o treino do modelo e, por isso teriam de receber algumas reclamações, atribuir uma categoria manualmente e só depois era possível automatizar este sistema.

5. Conclusão

Ao longo deste projeto foi possível implementar uma solução para a análise automatizada de reclamações, aproveitando os avanços na área da similaridade de texto. Ficou claro que a abordagem utilizada não só oferece uma solução viável para a análise de reclamações, como também aumenta a eficiência de uma empresa que utilize esta solução. A capacidade de automatizar a identificação de padrões nos feedbacks fornecidos pelos clientes economiza tempo e também proporciona novos pontos de vista para que melhorias sejam aplicadas na aplicação.

Ainda que os objetivos tenham sido atingidos é importante salientar que é possível melhorar a performance do algoritmo através de uma maior amostra de reclamações ou até mesmo a mudança para um algoritmo mais complexo dependendo da necessidade da entidade que o usar.

A realização deste projeto contribuiu para a assimilação destes novos conceitos de uma área que até então desconhecia fortalecendo o meu entendimento sobre a similaridade de texto e ampliando a minha capacidade de aplicar estes conceitos a outros projetos futuros.

6. Bibliografia

Knuth, D. (1973). *The Art of Computer Programming.* Adison Wesley.

Majid M., S. N. (2022). *Computing semantic similarity of texts based on deep graph learning with ability to use semantic role label information.*

Mingyu J., X. Z. (2022). *A Short Text Similarity Calculation Method Combining Semantic and Headword Attention Mechanism.*

Mohammadi, H. &. (2020). A fast text similarity measure for large document collections using multireference cosine and genetic algorithm. *28*(2), pp. 999-1013.

Nora, R., Moncy, R., P, R., Sreedarsana, A., & Sindhya, N. (s.d.). Sentence Similarity - A State of Art Approaches.

PennState University Libraries. (15 de Março de 2017). *APA Quick Citation Guide*. Obtido de PennState University Libraries Web Site: http://guides.libraries.psu.edu/apaquickguide/intext

Weidong Z., X. L. (2022). *Re-LSTM: A long short-term memory network text similarity algorithm based on weighted word embedding.*