



Universidade do Minho

MESTRADO INTEGRADO EM ENGENHARIA
INFORMÁTICA

INTELIGÊNCIA AMBIENTE: TECNOLOGIAS E APLICAÇÕES

INTELIGÊNCIA AMBIENTE COM SUPORTE DE
PROCESSAMENTO DE LINGUAGEM NATURAL

GRUPO 2



Diogo Monteiro **A83638**



Joana Gomes **A84912**



Maria Pires **A86268**



Susana Marques **A84167**

11 DE JANEIRO DE 2021

Conteúdo

1	Introdução	3
2	Abordagem de análise de sentimentos utilizada	4
3	Abordagem de chat escolhida para as Sugestões	6
4	Data Logging & Análise	8
4.1	Sentimentos	8
4.2	KeyboardInput	9
5	Demonstração	12
6	Conclusão	17

Lista de Figuras

1.1	Imagem do Chatbot desenvolvido	3
2.1	Exemplo de análise de sentimentos negativos	5
2.2	Exemplo de análise de sentimentos positivos	5
2.3	Exemplo de análise de sentimentos com negação	5
3.1	6
3.2	Janela das sugestões	7
3.3	Botão <i>refresh</i> (encontra-se no canto superior direito da janela principal da aplicação)	7
4.1	Botão para aceder aos gráficos (encontra-se à direita na janela principal do <i>chatbot</i>)	8
4.2	Sentimento guardado com o instante	9
4.3	Sentimentos ao longo do tempo	9
4.4	Input guardado	10
4.5	Percentagem das diferentes ações	10
4.6	Contador de teclas utilizadas	11
5.1	Janela inicial	12
5.2	Enviar uma mensagem	13
5.3	Resposta da Angie	13
5.4	Acéder à janela das sugestões	14
5.5	Fazer uma sugestão	14
5.6	Acéder à janela dos gráficos	15
5.7	Janela dos gráficos	15
5.8	Dar refresh	16

1 | Introdução

O presente relatório foi elaborado no âmbito da Unidade Curricular **INTELIGÊNCIA AMBIENTE: TECNOLOGIAS E APLICAÇÕES**, tendo como objetivo a conceção e aplicação de técnicas de Processamento de Linguagem Natural no contexto de ambientes inteligentes capazes de reconhecer e interpretar interações textuais.

Neste trabalho prático foi-nos proposta a construção de um **CHATBOT** com a capacidade de avaliar a emoção predominante no texto recebido e exibi-la em conjunto com a mensagem. O gestor do dialogo é responsável não só pela transformação da mensagem recebida numa resposta mas também por manter a ilusão de se tratar de uma conversa entre dois seres humanos [1]. Adicionalmente, o utilizador pode configurar um conjunto de respostas automáticas para perguntas comuns. Por fim, quando o *chat* é terminado o sistema apresenta um resumo da conversa.

Para a resolução do problema apresentado o grupo recorreu à linguagem **PYTHON**, usufruindo do poder das bibliotecas **NATURAL LANGUAGE TOOLKIT SPACY** e **PYNPOT**.

A personagem criada, **ANGIE**, dá vida ao amigo pronto a ajudar qualquer utilizador no combate à solidão e à tristeza, oferecendo apoio e soluções adequadas que permitem melhorar o estado de espírito do utilizador.

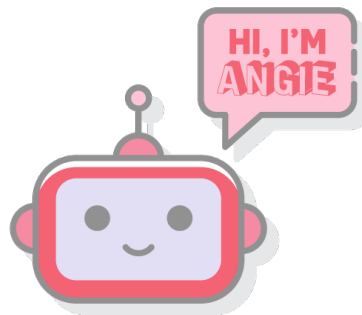


Figura 1.1: Imagem do Chatbot desenvolvido

2 | Abordagem de análise de sentimentos utilizada

Neste projeto foi-nos pedido que, para o *chatbot* criado pelo grupo, por cada mensagem enviada pelo utilizador, o sistema deverá avaliar a emoção predominante no texto e exibi-la com a mensagem. De forma a tornar o trabalho mais homogêneo, decidimos usar apenas a biblioteca *spaCy* em inglês, o que significa que todas as alterações feitas no texto vão ter como base a língua inglesa.

A análise de sentimentos foi feita com intuito de representar 3 estados emocionais: **Positivo, Neutro e Negativo**.

Posto isto, começamos por alterar o texto que nos é dado pelo Cliente, ou seja o texto que o utilizador insere na *text box* quando comunica com a *Angie*. Este texto passa primeiramente por uma alteração a nível de **processamento de linguagem natural**, quando usamos a biblioteca *spaCy* para realizarmos *tokenization*, ou seja, para separarmos cada palavra do texto.

A seguir, removemos "**stop words**", que são as palavras mais comuns e que não servem para obter o contexto de determinada frase, como por exemplo o "*the*". Sabemos que isto nos dá menos informação, mas o grupo decidiu optar por filtrar estas palavras de modo a obter uma maior rapidez na análise de sentimentos, tendo consciência que era insignificativo o valor que estas palavras tinham para obter um melhor contexto da situação em que a frase transparece.

Posteriormente, decidimos normalizar as palavras filtradas usando **lemmatization** (uma vez que a biblioteca *spaCy* não possui o método *stemming* e este já se tornou obsoleto) para agrupar palavras que não estejam na sua forma mais simples, como por exemplo verbos em tempos distintos (passado ou futuro), e as analisar apenas na forma mais simples.

Realizamos todas estas alterações, para depois conseguirmos fazer uma comparação mais eficaz entre as palavras que sofreram normalização e duas listas onde se encontram palavras consideradas positivas ou negativas (respetivamente). A lógica do grupo foi termos uma variável (***sentimental_rate***) que irá aumentar ou diminuir x unidades de acordo com a quantidade x de palavras que se encontram na frase que pertençam à lista de palavras positivas ($+x$) ou de palavras negativas ($-x$). Se o utilizador não inserir palavras que pertençam a nenhuma destas listas, então consideramos o seu estado como **Neutro**.

Por fim, surgiu um problema em relação a **frases** que estejam **na negativa**, uma vez que, pelo contexto, a emoção do utilizador será o oposto ao esperado se apenas observarmos o valor emocional de cada palavra. Para o resolvermos, usamos a estratégia de usar uma lista ("**nots**") onde se encontram as palavras

que em frases inglesas transformem o significado da frase para o oposto: ("no", "not", "nt", "n't", "cannot"). Aqui apenas usamos o método *tokenization*, uma vez que, se fizemos uma alteração mais profunda, a negação de verbos iria tornar-se apenas no verbo em si, e perder-se-ia o contexto. Usamos "nt" e "n't" de forma a representar todas os verbos na negação, já que ao recorrer ao *tokenization*, por exemplo, a palavra "dont" é transformada em "do" e "nt" e a palavra "don't" é transformada em "do" e "n't". Sabemos que "dont" não existe oficialmente na língua inglesa, mas é algo usual que informalmente as pessoas escrevem e facilmente poderá aparecer como resposta de um utilizador a um *chatbot*.

Neste trabalho tentamos observar, perceber e resolver todos estes pequenos problemas, pondo-nos no lugar de quem usaria um *chatbot* e como o faria.

Finalmente, percebemos que conseguíamos alterar o estado do utilizador para o oposto se percebemos que ele estava a usar frases na negativa, alterando a nossa flag (*sentimental_rate*) para valores bastante grandes ou bastantes pequenos, caso a frase antes estivesse na negativa ou na positiva, respetivamente.

A seguir encontram-se exemplos explicativos de como o nosso programa se comporta e analisa os sentimentos do utilizador quando ele insere uma frase com conetividade negativa ou positiva e ainda na presença de uma negação.

```
User: Hello, how are you? (Neutral)
Angie: I'm okay right now, how about you?
User: I'm fine :) (Positive)
Angie: Alright, I'm glad you are feeling okay!
```

Figura 2.1: Exemplo de análise de sentimentos negativos

```
User: Hello, how are you? (Neutral)
Angie: I'm okay right now, how about you?
User: I'm a little sad today? (Negative)
Angie: I'm here to cheer you up! Talk to me :)
```

Figura 2.2: Exemplo de análise de sentimentos positivos

```
Angie: Hello! I'm Angie, how can i help you?
User: I don't feel happy... (Negative)
Angie: I'm here for you, what's wrong?
```

Figura 2.3: Exemplo de análise de sentimentos com negação

3 | Abordagem de chat escolhida para as Sugestões

O grupo decidiu criar um *chatbot* (incorporando um *socket* entre um *Server* e um *Client*) cujo objetivo é ajudar pessoas que se sentem solitárias em tempos de *Covid* e, por exemplo, no Natal, que é um feriado com grande simbologia familiar.

O nosso *chatbot* consiste em diferentes respostas para uma série de perguntas que o utilizador pode fazer adequadas ao tema referido. Estas respostas ou sugestões que a *Angie* emite são seleccionadas de forma aleatória, depois de treinarmos um modelo com diversos padrões que nos permite responder acertadamente, com alta probabilidade ao que é pedido pelo utilizador, mantendo uma conversa o mais estável possível.

No treino do modelo, usamos novamente *processamento de linguagem natural* para ajudar a prever mais facilmente o resultado correto e optamos por classificar cada resposta com uma determinada probabilidade e posteriormente ordenamos essas probabilidades para que a escolha mais certa no final seja a que tem maior probabilidade perante o modelo treinado.

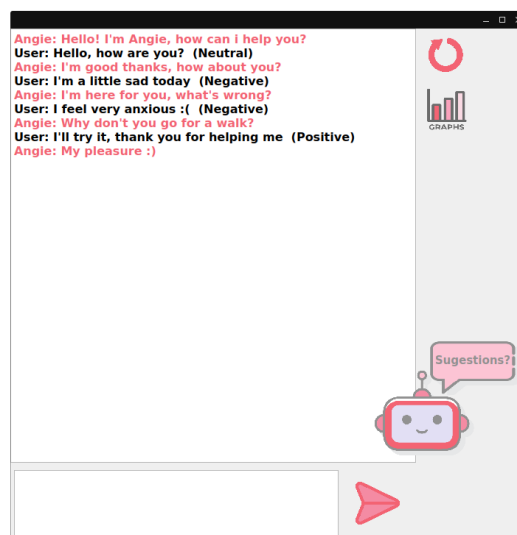


Figura 3.1

Os dados deste modelo encontram-se no ficheiro *intents.json* que pode ser alterado e editado em qualquer altura desde que depois se faça novamente o treino do modelo.

Uma forma eficiente para que estas sugestões não se encontrem implementadas com "força bruta" ou seja, fazendo o utilizador escrever realmente no ficheiro, foi acrescentarmos ao programa uma secção que permite que este digite as suas sugestões e estas serem implementadas em tempo real no ficheiro *intents.json* sem que o utilizador tenha que entrar em contacto com este ficheiro.

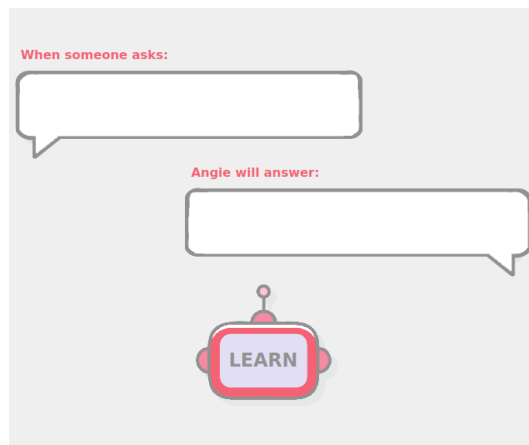


Figura 3.2: Janela das sugestões

Finalmente, basta o utilizador usar o botão de *refresh* que irá obrigar o programa a treinar o modelo outra vez e reiniciar a conversa com a *Angie*, tornando o o modelo e o programa cada vez mais rico de acordo com a informação que se treina.



Figura 3.3: Botão *refresh*

(encontra-se no canto superior direito da janela principal da aplicação)

4 | Data Logging & Análise

Nesta secção do nosso trabalho, visamos desenvolver métodos de captação de dados gerados no decorrer da utilização do programa. Para tal, tivemos que considerar todo o tipo de informação capaz de ser gerada pelo utilizador, tal como a necessidade da captura da mesma, ou seja, tomar juízos de valor sobre que informação será do nosso interesse capturar. Posteriormente, estes dados serão extremamente úteis tanto para análise como para um conjunto de funções que iremos mencionar.

Tendo debatido sobre o tópico e de forma a cumprirmos os requisitos estabelecidos pelo enunciado, os valores que consideramos importantes de salvar foram os seguintes:

- **Sentimentos**
- **Keyboard Input**

Ambos estes tipos de dados, tal como a informação auxiliar que julgamos ser necessária, serão guardados em ficheiros *txt* específicos.

É possível aceder ao menu dos gráficos descritos nesta secção através do botão **GRAPHS**.



Figura 4.1: Botão para aceder aos gráficos
(encontra-se à direita na janela principal do *chatbot*)

4.1 Sentimentos

Os sentimentos expressidos pelo utilizador em cada mensagem que escreve são, evidentemente, dados de extrema importância e, por consequente, foram os primeiros que decidimos guardar.

Tal como mencionado previamente, esta informação é utilizada pelo nosso bot de forma a gerar respostas de apoio emocional, adequadas ao estado de espírito do nosso utilizador. Porém, nesta secção do trabalho, a sua captação servirá para visualização estatística.

Guardamos esta informação quando o utilizador envia a sua mensagem para o bot e fica guardada em conjunto com o instante de tempo do envio.

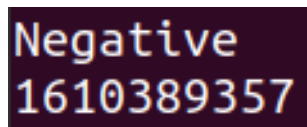


Figura 4.2: Sentimento guardado com o instante

O utilizador poderá carregar no botão da visualização dos gráficos e será apresentado com um gráfico/resumo da variação do seu humor ao longo da conversa:

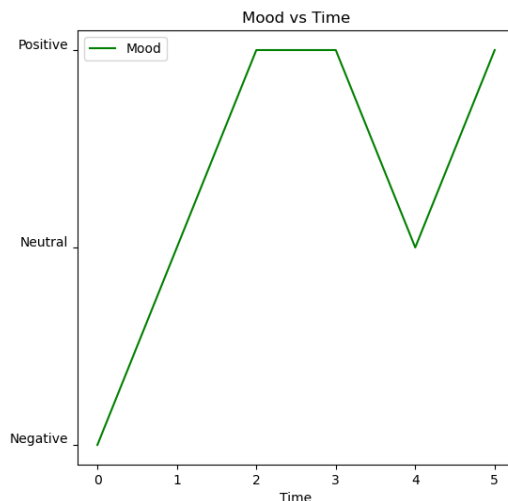


Figura 4.3: Sentimentos ao longo do tempo

4.2 KeyboardInput

No decorrer da utilização do programa, o utilizador vai realizar diversas ações que requerem inputs no teclado. Seja quando está a escrever uma mensagem, ou então, caso seja necessário, a deleção dos caracteres que acabou de escrever, por exemplo.

A captura da informação sobre as teclas que o utilizador utiliza em conjunto com o momento em que a tecla é carregada e largada, será bastante interessante, tanto no ponto de vista estatístico, como para uma avaliação psicológica do utilizador.

Para além disso, a ideia e posterior decisão de captarmos e utilizarmos estes dados é proveniente do estudo sobre estes mecanismos ao longo do semestre. Sendo que, todas as funções que demos uso para a captura dos inputs do teclado, são oferecidas pela biblioteca **pynput** apresentada durante as aulas.

Neste momento, os valores de input do teclado, que o nosso programa captura, são os seguintes:

```
alphanumeric key a released  
1610389390  
alphanumeric key v pressed  
1610389390  
alphanumeric key e pressed  
1610389390  
alphanumeric key v released  
1610389390  
alphanumeric key e released  
1610389390
```

Figura 4.4: Input guardado

Depois da sua captura e com o auxílio de algum processamento, podemos fazer demonstrações estatísticas da percentagem de vezes que o utilizador realiza toques rápidos, lentos e até a quantidade de deletions realizadas. Não sendo um método completamente científico, podemos interpretar um utilizador cuja taxa de deleções e toques rápidos seja muito elevada, como estando com um nível de stress elevado.

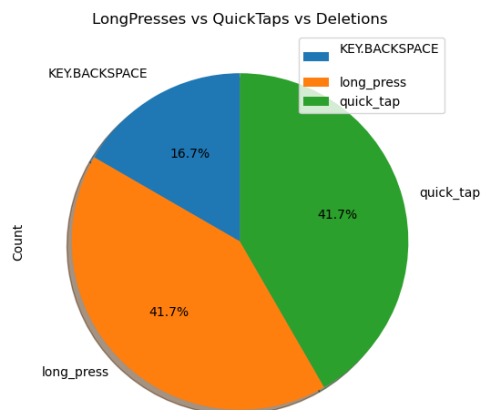


Figura 4.5: Percentagem das diferentes ações

Para além disso, também somos capazes de apresentar um gráfico que demonstra as teclas que o utilizador utilizou em conjunto com a sua contagem.

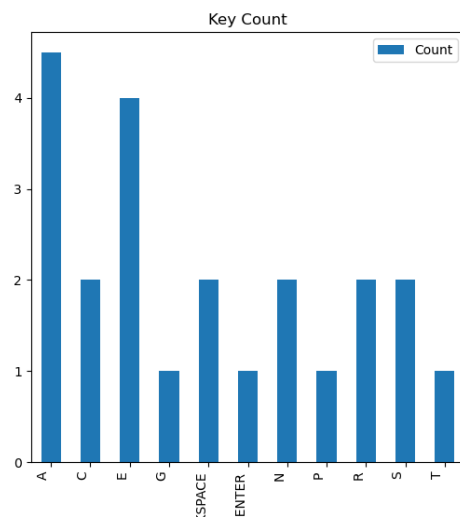


Figura 4.6: Contador de teclas utilizadas

5 | Demonstração

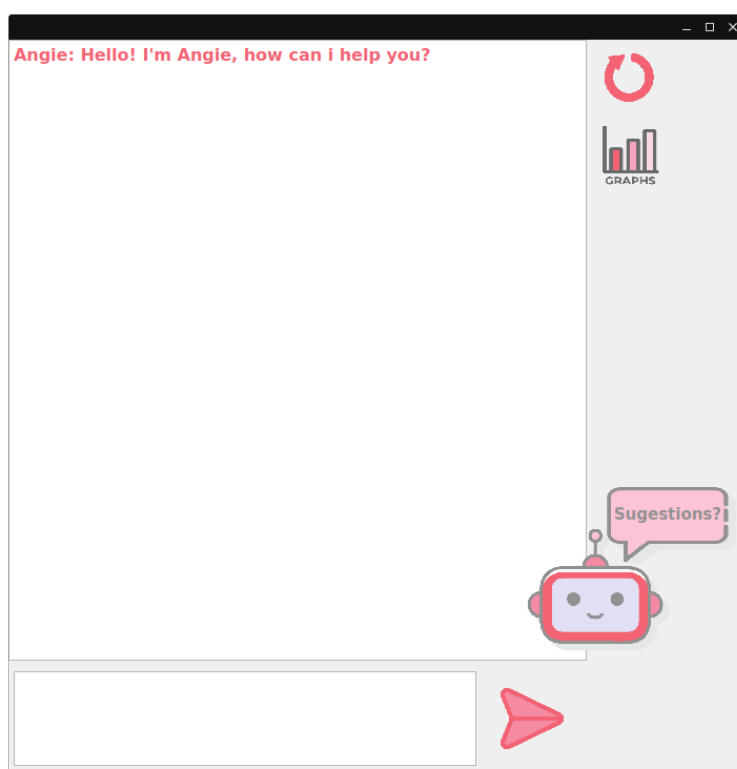


Figura 5.1: Janela inicial



Figura 5.2: Enviar uma mensagem

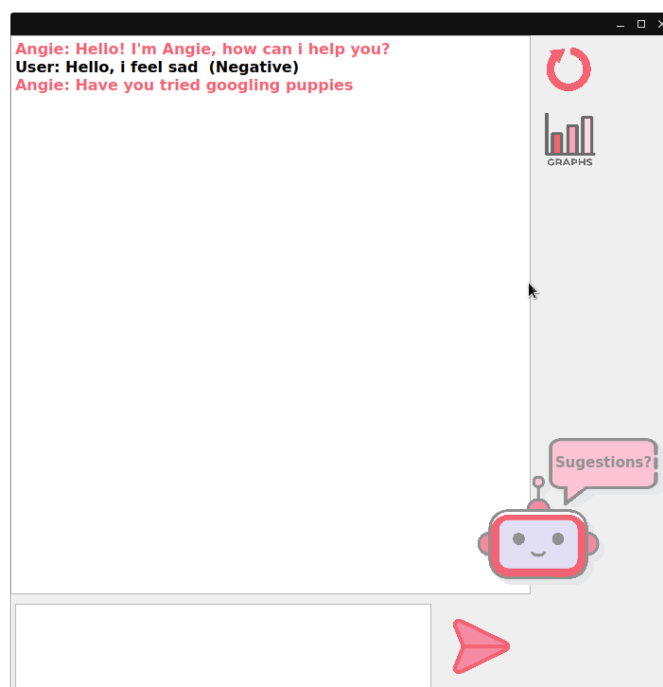


Figura 5.3: Resposta da Angie

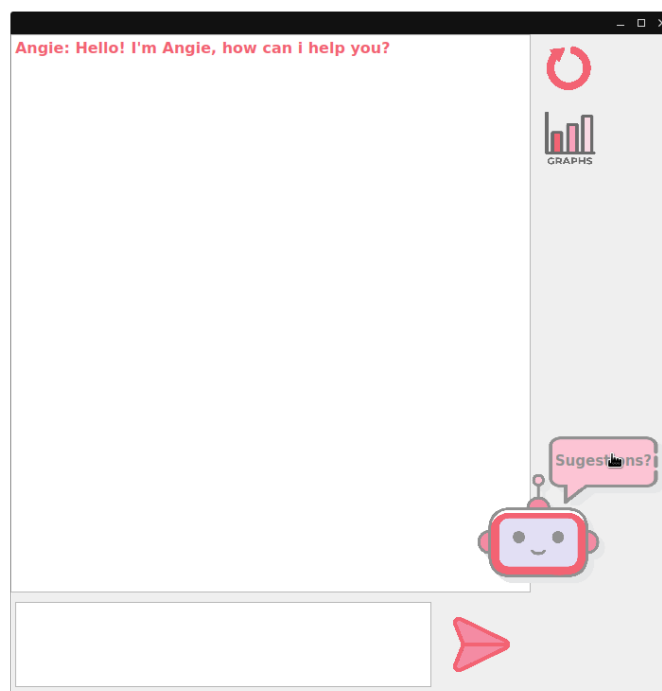


Figura 5.4: Aceder à janela das sugestões

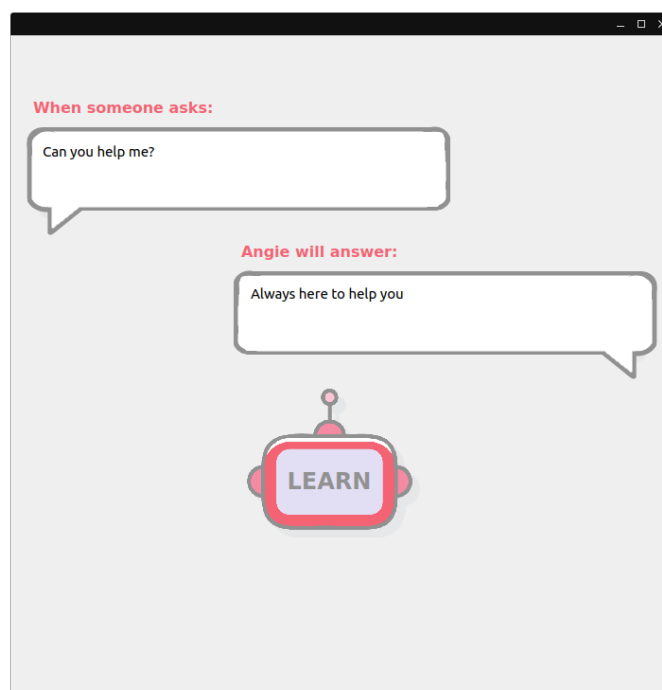


Figura 5.5: Fazer uma sugestão

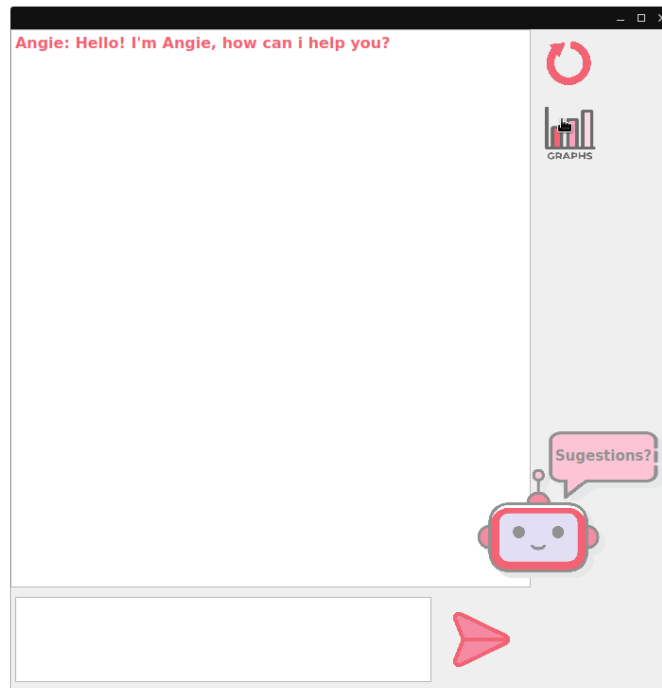


Figura 5.6: Aceder à janela dos gráficos

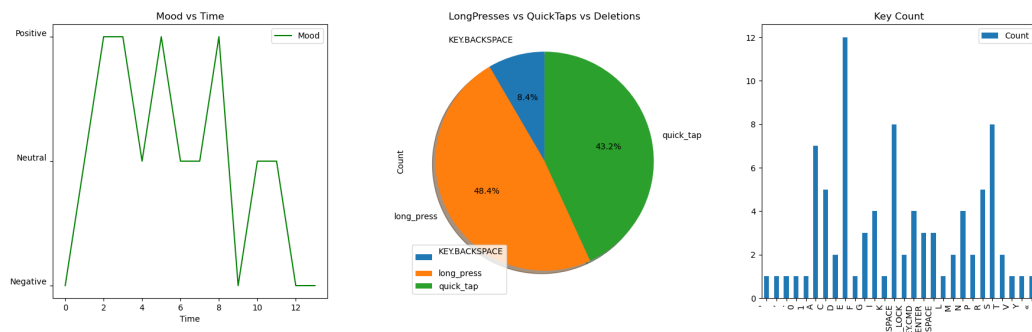


Figura 5.7: Janela dos gráficos

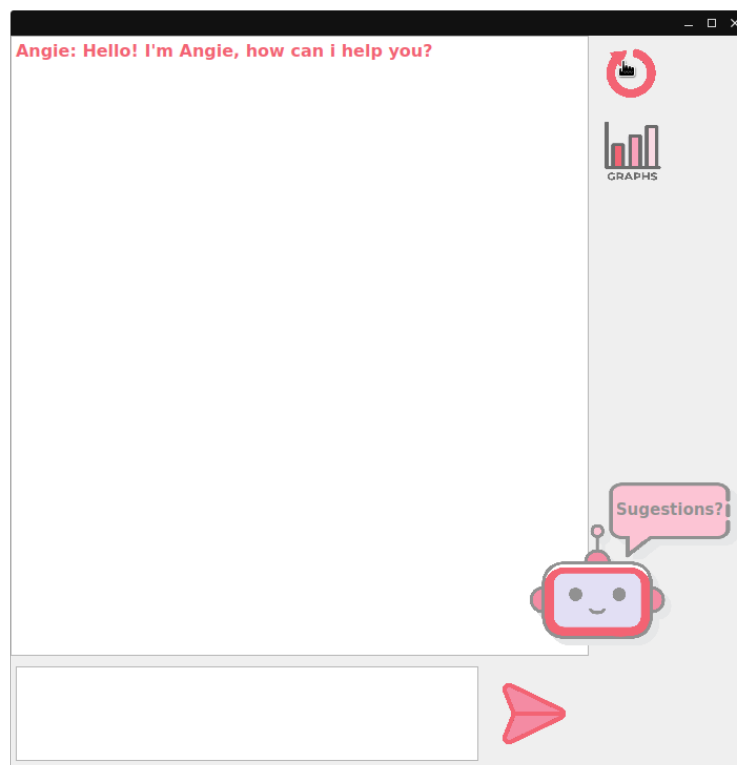


Figura 5.8: Dar refresh

6 | Conclusão

Em suma, os objetivos para a realização deste trabalho foram cumpridos, tendo sido implementado um *chatbot* que contempla todas as funcionalidades requeridas.

A realização deste projeto dotou-nos da capacidade de aplicação e manipulação de técnicas de Processamento de Linguagem Natural no tratamento de interações textuais no contexto de ambientes inteligentes.

Bibliografia

- [1] Marcondes, F. S., Almeida, J. J., Novais, P. (2018, November). Chatbot Theory. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 374-384). Springer, Cham.
- [2] Natural Language Toolkit — NLTK 3.5 documentation. Nltk.org. (Published 2020). Accessed December 22, 2020. <https://www.nltk.org/>
- [3] Pynput. PyPI. (Published December 21, 2020). Accessed January 3, 2021. <https://pypi.org/project/pynput/>