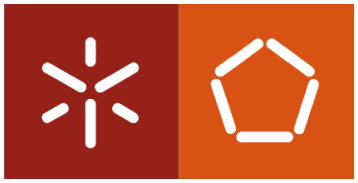


Universidade do Minho
Mestrado em Engenharia Informática
AAII

Criação de Embeddings Moleculares Baseados em Algoritmos de NLP para Integração na Ferramenta DeepMol

Gonçalo Almeida - A84610
Pedro Ribeiro - PG42848
Raimundo Barros - PG42814

Professor Doutor Miguel Rocha

Tutor
João Filipe Silva Correia

Mestrado em Engenharia Informática
2021

Abstract

A aplicação de abordagens de *Machine Learning* (ML) e *Deep Learning* (DL) em tópicos de quimioinformática tem como objetivo a identificação e o *design* de moléculas com propriedades específicas. Técnicas baseadas em modelos de processamento de linguagem natural (NLP) são capazes de criar *embeddings* moleculares a partir de representações textuais como SMILES. Portanto, o projeto passa pela implementação e integração de abordagens baseadas em algoritmos de NLP, tais como *SMILES Transformer*, Mol2vec e Seq2seq para a criação de *embeddings* moleculares de modo a serem utilizadas em modelos de classificação de compostos na ferramenta DeepMol. As tarefas realizadas visam a exploração da *pipeline* existente usando o *dataset* HIV.

Keywords: *Machine Learning*, *Deep Learning*, NLP, *Embeddings* Moleculares, SMILES, DeepMol.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Enquadramento

Este trabalho foi proposto pelo docente da unidade curricular com o objectivo de desenvolver abordagens de *Machine* e *Deep Learning* na óptica da quimioinformática onde o objectivo passa pela identificação e design de moléculas com propriedades específicas. Foram sugeridos vários algoritmos onde o grupo foi capaz de explorar três: Mol2Vec, SMILES Transformer e Seq2Seq2 Fingerprint. Para além da exploração dos algoritmos a sua implementação no DeepMol é um processo fundamental para a evolução da ferramenta.

1.2 Fluxo de Trabalho

O *workflow* que seguimos para a realização deste projeto seguiu o seguinte formato:

- Familiarização com a ferramenta DeepMol;
- Estudo, análise e seleção dos algoritmos sugeridos;
- Criação de diversos *notebooks* para uma implementação preliminar dos algoritmos;
- Criação de módulos para integração com o DeepMol.

1.3 Estrutura do Relatório

Os tópicos que iremos cobrir ao longo deste relatório encontram-se distribuídos da seguinte forma:

- Capítulo 2: Descrição do contexto tecnológico e das abordagens exploradas;
- Capítulo 3: Análise dos algoritmos utilizados, desenvolvimento e implementação no DeepMol;
- Capítulo 4: Conclusões e trabalho futuro.

Capítulo 2

Contexto e Ferramentas

2.1 *Embeddings* Moleculares

Baseadas em *Word Embeddings*, com *embeddings* moleculares, as moléculas são representadas por vetores densos, onde um vetor representa a projeção da molécula num espaço vetorial. A posição de uma molécula dentro do espaço vetorial é relativa ao posicionamento no SMILES.[1]

2.1.1 SMILES

Tipicamente as moléculas têm uma representação química porém, para poderem ser representadas através de caracteres ASCII, foi criada a representação *Simplified Molecular Input Line Entry Specification* ou simplesmente SMILES. Esta representação textual visa criar uma estrutura onde todas as moléculas podem ser representadas em formato texto. Tendo como exemplo o Dióxido de Carbono, com a fórmula química CO_2, a sua representação em SMILES passa a ser: O=C=O.

2.2 DeepMol

DeepMol é uma *framework* de *Deep Learning* desenvolvida na linguagem de programação Python para a análise e estudo de compostos químicos. Esta *framework* engloba um conjunto de módulos para a realização de *embeddings* moleculares, tendo como base a implementação de métodos que sejam capazes de criar representações vetoriais que caraterizem substruturas moleculares. Para a construção de modelos de ML e DL, o *DeepMol* faz uso de *packages* como Tensorflow, Keras, Scikit-learn, RDKit e DeepChem. Com a implementação dos algoritmos Mol2vec, Seq2seq Fingerprint e SMILES Transformer enriquecemos ainda mais a ferramenta em termos de opções para a *featurization* de compostos.

2.3 Mol2vec

Mol2vec é uma abordagem de aprendizagem máquina não supervisionada para aprender representações vetoriais de subestruturas moleculares. Como os modelos Word2vec, onde vetores de palavras intimamente relacionadas estão em estreita proximidade no espaço vetorial, Mol2vec aprende representações vetoriais de subestruturas moleculares que apontam em direções semelhantes para subestruturas quimicamente relacionadas. Os compostos podem finalmente ser codificados como vetores pela soma dos vetores das subestruturas individuais para, por exemplo, serem alimentados em abordagens de aprendizagem máquina supervisionadas para prever propriedades dos compostos. Os *embeddings* são obtidos com o treino de um modelo num corpus de compostos que consiste em toda a matéria química disponível. Os recursos de previsão são demonstrados em vários conjuntos de dados de propriedades e bioatividade de compostos e comparados com os resultados obtidos para impressões digitais de Morgan como uma representação de composto de referência.

2.4 Seq2Seq Fingerprint

O Seq2seq ou *Sequence-to-sequence* é uma abordagem utilizada por modelos de *Machine* e *Deep Learning* para modelos de aprendizagem não supervisionada que recebem um determinado *input* e convertem a sequência para uma representação num outro domínio. Ao longo dos anos esta abordagem é a mais utilizada para problemas de tradução entre linguagens, mais propriamente entre o Inglês e o Francês (eg. “le chat est noir” -> [Seq2Seq] -> “the cat is black”). As *fingerprints* são camadas de *features* que aplicam uma função de *hash* estática com a concatenação das *features* da camada anterior. Através da utilização de um modelo LSTM, a rede é capaz de construir um significado para o respetivo SMILES. Embora este método seja conhecido pelo seu treino lento, o tempo investido traz mais valias para o modelo.

2.5 SMILES Transformer

SMILES Transformer é uma abordagem baseada em Transformers e modelos pré-treinados para NLP e consiste na aprendizagem de *fingerprints* moleculares através de um pré-treinamento não supervisionado de um modelo de linguagem *sequence-to-sequence*. Assim sendo, codifica as representações textuais dos *fingerprints* moleculares para representações vetoriais.

Capítulo 3

Exploração e Integração

3.1 Dataset

O *dataset* HIV utilizado para todas as abordagens é constituído por três colunas: *SMILES*, *Activity* e *HIV_Active* para 41127 amostras, onde a segunda coluna é constituída por um dos elementos de um conjunto de perfis de atividade { CM, CI, CA } e a última coluna é um valor binário {0,1} que indica se o vírus HIV está ativo.

Foi realizada uma análise da distribuição da classe *HIV_Active* (ver Fig. 3.1) para perceber o balanceamento dos dados relativamente à classe que queremos classificar, e obtivemos os seguintes resultados:

- 0 (Negativo): 39684
- 1: (Positivo): 1443

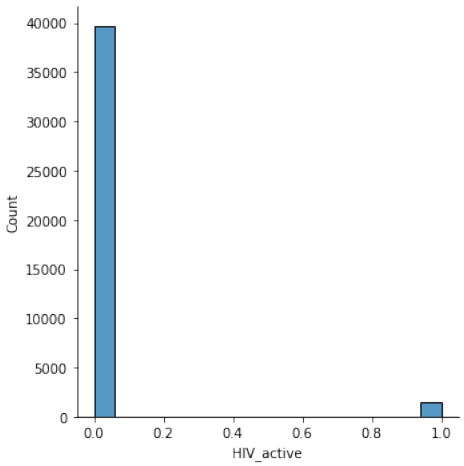
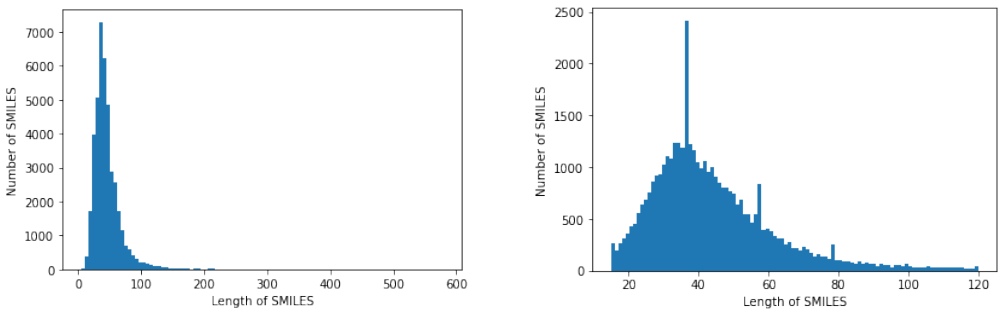


Figura 3.1: Distribuição dos dados para HIV Positivo e Negativo

Analisamos também a distribuição dos comprimentos dos SMILES e detetamos a presença de vários *outliers* que poderiam causar-nos problemas nas implementações dos algoritmos. Definimos um intervalo de [15, 120] para limitar o número de *tokens* do vocabulário dos SMILES, sendo que apenas perdemos 2.03% das amostras iniciais.



(a) *Dataset* inicial (b) *Dataset* reduzido

Figura 3.2: Distribuição dos comprimentos dos SMILES

3.2 Mol2vec

3.2.1 Abordagem

Mol2vec possui comandos para preparar um corpus a partir de dados moleculares (ex: os *SMILES*), treinar o modelo Mol2vec e caraterizar novas amostras. Este gera identificadores Morgan que representam palavras (moléculas são sequências de palavras). As palavras são ordenadas na frase de acordo com a ordem dos átomos em SMILES canónicos.

Geração de moléculas para cada SMILES no *dataset*:

```
> molecules = [Chem.MolFromSmiles(x) for x in dataset.mols]
```

Mol2vec é baseado no algoritmo Word2vec, dessa forma, primeiro temos que codificar as moléculas como frases, o que significa que cada subestrutura (representada pelo identificador Morgan) representa uma palavra.

```
> sentence = mol2alt_sentence(mol[1], 1)
```

```
> sentence = ['2246703798', ... , '3218693969', ... , '4142616092']
```

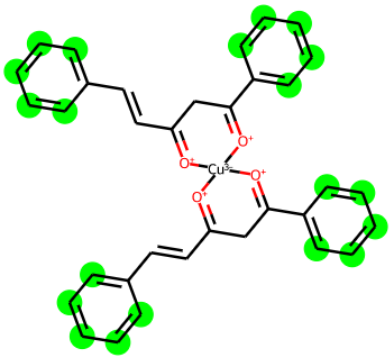


Figura 3.3: Representação de um identificador Morgan específico

3.2.2 *Featurization* no DeepMol

Para realizar a *featurization* dos compostos através do *framework* DeepMol, primeiramente deve-se carregar o ficheiro do *dataset* HIV utilizando o *CSVLoader*, onde é informado o campo das moléculas (no nosso caso foi a coluna dos *smiles*), e também informado o campo referente às *labels*.

```
> dataset = CSVLoader(dataset_path='HIV.csv',
                      mol_field='smiles',
                      labels_fields='HIV_active'))
> dataset = dataset.create_dataset()
```

As *features* das moléculas são criadas usando um modelo Mol2vec pré-treinado em 20 milhões de compostos e resultam em 300 *embeddings*.

```
> model = word2vec.Word2Vec.load('/content/model_300dim.pkl')
```

A partir do modelo pré-treinado e, com o *dataset* carregado, o mol2vec é executado seguindo as seguintes etapas:

1. gera as moléculas para cada elemento SMILES do dataset;
2. cada molécula gerada é codificada como uma frase;
3. gera vetores para cada frase numa lista de frases. O vetor é simplesmente uma soma de vetores para indivíduos palavras;
4. uma classe auxiliar é aplicada para armazenar os vetores num pandas *DataFrame*.

Ao rodar a *featurization* utilizando o modelo pré-treinado obtivemos o seguinte *shape*.

- Mols_shape: 41127
- Features_shape: (41127, 300)
- Labels_shape: (41127,)

3.2.3 Análise dos Resultados Obtidos

Com o objetivo de avaliar o desempenho dos resultados obtidos da *featurization* utilizando o modulo Mol2vec, o *dataset* gerado foi carregado em um modelo *Random Forest* para realizar a classificação dos dados obtidos.

Primeiramente o *dataset* foi dividido em treino (60%), validação (20%) e teste (20%). Após a divisão foi aplicada uma *cross validation* do modelo sobre o *dataset*. Feito o treino do modelo, foram obtidos os seguintes resultados:

	Score
AUC-ROC	0.58360
Precison	0.63636
Accuracy	0.96765

Tabela 3.1: *Scores* do modelo para os dados de teste

	Positivo	Negativo
Positivo	49	238
Negativo	28	7909

Tabela 3.2: Matriz de confusão do modelo para os dados de teste

classification_report:

	precision	recall	f1-score	suppor
0	0.97	1.00	0.98	7937
1	0.64	0.17	0.27	287
accuracy			0.97	8224
macro avg	0.80	0.58	0.63	8224
weighted avg	0.96	0.97	0.96	8224

Figura 3.4: Avaliação do modelo para os dados de teste

Com base nos resultados apresentados na Fig. 3.4, podemos observar que o modelo foi bem treinado, e que no geral possui uma boa *accurary* (97%). Contudo podemos observar que o valor do *recall* para a classe 1 (HIV positivo) é relativamente baixo (17%). Isso pode ser justificado devido a desbalanceamento da variável resposta, em que mais de 96% dos resultados do *dataset* se referem a amostras de HIV negativo.

3.3 Seq2Seq Fingerprint

3.3.1 Abordagem

O modelo começa por vetorizar os SMILES para um array. Para isto, um conjunto de caracteres é construído a partir de todos os caracteres encontrados nos SMILES (os de teste e de treino). Além disso, o caractere "!"é adicionado ao início do SMILES e completado com o caractere "E"até ao tamanho máximo do SMILES, desta forma é sinalizado o início e o fim do mesmo. Este processo pode ser observado na figura B.1.

Posteriormente um conjunto de caracteres e os dicionários são usados para definir os *bits* necessários na matriz em numpy, o que irá resultar num *Dummy* para cada SMILES, a figura B.2 é a representação gráfica.

Através do uso de LSTM e de camadas densas foi criado um para *Enconder-Decoder*, com uma camada única de 128 LSTM *cells* é utilizada para a leitura dos SMILES. O *outputs* são ignorados, porém o resultado interno final do estado C e H é concatenado e recombinado para uma *bottleneck layer* "neck". Para que o resultado do "neck"seja útil este é passa por duas camadas densas diferentes para fazer o *decode* dos estados que deram ser declarados no *decoder* da camada do LSTM. Após o treino do *autoencoder* podemos ver que a reconstrução dos SMILES não é 100% correta, devido à taxa de *loss* que, em média, ronda os 26%.

Infelizmente não foi realizada a integração deste algoritmo com o DeepMol pois a abordagem tomada não foi completamente compatível, tanto com a ferramenta, como com o problema em questão.

3.4 SMILES Transformer

3.4.1 Abordagem

Para a extração de *embeddings* com um SMILES Transformer consideramos o transformer BERT.

BERT (*Bidirectional Encoder Representations from Transformers*), lançado em 2018 por Jacob Devlin e os seus colaboradores da Google, é um modelo de representação de linguagem pré-treinado numa grande quantidade de textos não rotulados. Pode ser utilizado para extrair *features* de alta qualidade de dados textuais ou *fine-tuned* para tarefas específicas, como a classificação de textos, produzindo previsões do estado da arte. Visto que o nosso objetivo é apenas extrair *features* (*embeddings*) das representações moleculares textuais SMILES, não iremos afinar o modelo BERT ao nosso conjunto de dados.

Este processo de extração de *features* irá seguir o seguinte *workflow*:

- Inicializar um modelo base e carregar pesos pré-treinados para o mesmo;
- Correr o nosso *dataset* pelo modelo e guardar o *output* de uma ou mais das suas camadas (este processo é denominado de *Feature Extraction*);
- Utilizar estes *outputs* como dados de *input* num novo modelo (este processo é denominado de *Transfer Learning*).

Começamos por inicializar o modelo e *tokenizer* 'bert-base-cased' da Hugging Face pois fornece uma interface para o Tensorflow 2 para trabalhar com o modelo BERT. Escolhemos a versão *cased* do modelo visto que os *tokens* no vocabulário da representação SMILES são sensíveis a maiúsculas, por exemplo, o token para o elemento químico Carbono 'C' deve ser diferenciado do token para o elemento aromático 'c'.

De modo a formatar os dados para servirem de *input* ao modelo é necessário realizar uma tokenização de cada sequência de *tokens* ou SMILES. Este processo devolve um vetor em que cada elemento é o índice de cada *token* no vocabulário do *tokenizer*. Há que notar que os *tokens* que não existem no vocabulário são divididos em *tokens* presentes no mesmo, e são adicionado dois índices de *tokens* reservados, um que representa o início da sequência (101) e outro que representa o fim (102). Ainda, é realizado um processo de *padding* para garantir que todas as sequências são do mesmo comprimento.

A seguir é demonstrado este processo de tokenização.

Sequência de *tokens*:

C C C 1 = [O +] [Cu − 3] 2 ([O +] = C (C C) C 1)

Índices dos *tokens* no vocabulário:

```
[ 101  140  140  140  122  134  164  152  116  166  164  140 1358
  118  124  166  123  113  164  152  116  166  134  140  113  140
  140  114  140  122  114  102 ]
```

O Transformer BERT contêm 12 *hidden states* correspondentes aos *outputs* das suas 12 *hidden layers*. Como prática mais comum na extração de *embeddings* com o BERT, iremos considerar o *output* da última camada. Contudo, cada *token* é representado por um vetor com 768 *features*, o que significa que cada sequência de *tokens* é representado por uma matriz com dimensões 'comprimento das sequências x 768'. Como o nosso objetivo é representar cada SMILES com um vetor uni-dimensional, a estratégia utilizada foi calcular a média dos valores de cada *token*. Desta forma, cada SMILES passa a ser representado por um vetor uni-dimensional com 768 *features*.

Todos os passos desta abordagem foram explicados e demonstrados no *notebook* denominado *bert_ Explanation.ipynb* que se encontra na diretoria de testes dos Transformers.

3.4.2 Análise dos Resultados Obtidos

Como forma de comparar o desempenho da *featurization* desta abordagem com as abordagens anteriores, foi aplicado um modelo Random Forest para realizar a classificação dos dados.

O *dataset* foi dividido em conjuntos de treino, de validação e de teste com as respetivas percentagens 60%, 20% e 20% dos dados. A performance do modelo foi validada aplicando a técnica de *cross-validation* com 3 *folds*.

Após o treino do modelo, para os dados de teste, foram obtidos os seguintes resultados:

	Score
AUC-ROC	0.54227
Precison	0.69697
Accuracy	0.96840

Tabela 3.3: *Scores* do modelo para os dados de teste

	Positivo	Negativo
Positivo	23	245
Negativo	10	7792

Tabela 3.4: Matriz de confusão do modelo para os dados de teste

```
classification_report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      7802
           1       0.70      0.09      0.15       268

    accuracy                           0.97      8070
   macro avg       0.83      0.54      0.57      8070
weighted avg       0.96      0.97      0.96      8070
```

Figura 3.5: Avaliação do modelo para os dados de teste

Com base nos resultados apresentados, pode-se observar que o valor do *score AUC-ROC* é próximo de 0.5, o que significa que o modelo apresenta algumas dificuldades a distinguir as classificações positivas das negativas, o que pode ser justificado pelo desbalanceamento dos dados relativamente à classe *HIV_Active*.

O valor da *Accuracy* foi alto, contudo, devido ao desbalanceamento, isto é espectável. Um modelo que classifique todas as instâncias com a classe predominante também terá um valor de *Accuracy* alto logo, esta não será uma boa métrica para avaliar o modelo.

Com a *Precision*, por outro lado, podemos avaliar a percentagem de casos que foram classificados como positivos corretamente, 70% dos casos que foram classificados como positivos estão corretos.

Analisando agora o valor de *Recall*, para os casos positivos, apenas 9% foram corretamente classificados. Visto que este se trata de um problema ligado à medicina, classificar apenas 9% dos pacientes com uma doença que estes efetivamente têm não é de todo ideal.

Os resultados obtidos não são ideais para o problema em questão, sendo que o desbalanceamento dos dados é uma das principais causas para tal. Um teste que poderá ser realizado seria diminuir ainda mais o comprimento máximo das sequências de *tokens*, visto que este fator também pode ser uma fonte de problemas para classificadores com *embeddings*.

Estes resultados apresentados foram obtidos com a integração das tarefas realizadas no *notebook bert_Explanation.ipynb* para a ferramenta DeepMol. Foi desenvolvido um módulo Python denominado *bert.py* no *package compoundFeaturization* que é utilizado para realizar a *featurization* dos SMILES para SMILES Embeddings através do modelo BERT. Há que referir 3 *notebooks* presentes na secção dos testes dos Transformers, o *prepare_data.ipynb* que contêm uma análise e pré-tratamento dos dados, o *bert_EmbeddingsExtraction.ipynb* que contêm a aplicação da abordagem descrita e, por fim, o *bert_RandomForest.ipynb* que contêm a avaliação do modelo Random Forest analisado.

Capítulo 4

Conclusões

O objetivo deste trabalho consistiu na análise de dados num dado contexto de aplicação, no nosso caso, a criação de *embeddings* moleculares para compostos químicos.
Apesar de ter consumido uma quantidade de tempo significativa, o estudo da *framework* DeepMol demonstrou-se muito importante e facilitou a fase de integração das nossas abordagens na ferramenta.
Um dos maiores desafios encontrados no decorrer deste projeto foi a escolha dos algoritmos e a interpretação dos mesmos para serem aplicados ao nosso contexto.
Consideramos que a implementação dos algoritmos e a integração no DeepMol foi realizada com sucesso para os algoritmos Mol2Vec e SMILES Transformer, contudo, os resultados não foram ideais. Percebemos que podia-mos ter realizado alguns ajustamentos no tratamento dos dados e nos algoritmos para terem sido obtidos melhores resultados. Por outro lado, o facto de não termos conseguido integrar o Seq2Seq na *framework* é um ponto negativo do nosso projeto.
Como trabalho futuro seria interessante explorar ainda mais o DeepMol e tirar partido dos vários modelos que este disponibiliza.

Apêndice A

Proposta Original do Projeto

Proposta A. 2 - Desenvolvimento de abordagens para a criação de embeddings moleculares baseados em algoritmos de NLP para integração numa ferramenta de machine learning existente (DeepMol).

Os resultados alcançados, nas últimas décadas, pelo uso de abordagens de inteligência artificial em áreas como visão computacional e processamento de linguagem natural levaram a um uso mais generalizado dessas abordagens noutras áreas.

Um desses casos passa pela aplicação de aborgadens de Machine e Deep Learning em tópicos de quimioinformatica onde o objetivo passa pela identificação e design de moléculas com propriedades específicas. Para isto, o processo de implementação de métodos que sejam capazes de criar representações vetoriais que caracterizem substruturas moleculares assume grande importância.

Hoje em dia existem múltiplas técnicas baseadas em modelos de NLP capazes de eficientemente criar embeddings moleculares a partir de representações como SMILES. Exemplos disso são os algoritmos Mol2vec, Seq2seq Fingerprint, SMILES Transformer, Message Passing Neural Networks for SMILES, SMILES-BERT entre muitos outros. Desta forma, o objetivo deste projeto passa pela implementação e integração de um módulo de abordagens baseadas em algoritmos de NLP para a criação de embeddings moleculares numa ferramenta de machine learning de classificação de compostos.

Inicialmente, as tarefas passariam por explorar a pipeline existente usando um dataset como case study***. Depois de perceber a estrutura e organização da pipeline iria proceder-se à implementação e integração do módulo para criação de embeddings moleculares. O projeto será desenvolvido usando a linguagem Python.

Dataset	Data Type	Task Type	Compounds	Link
HIV: Experimentally measured abilities to inhibit HIV replicattion	SMILES	Binary Classification	41127	https://github.com/GLambard/Molecules_Dataset_Coll

Apêndice B

Imagens de auxilio

B.1 Seq2Seq Fingerprint

```
SMILES:
Cl.Cn1cc(NC(=O)c2cc(NC(=O)CCCCCCCCCCC(=O)Nc3cc(C(=O)Nc4cc(C(=O)NCCC(=N)N)n(C)c4)n(C)c3)cn2C)cc1C(=O)NCCC(=N)N
Pré-Dummy SMILES
!Cl.Cn1cc(NC(=O)c2cc(NC(=O)CCCCCCCCCCC(=O)Nc3cc(C(=O)Nc4cc(C(=O)NCCC(=N)N)n(C)c4)n(C)c3)cn2C)cc1C(=O)NCCC(=N)NEEEEEEEEEEEEEE
```

Figura B.1: Resultado de um Pré-Dummy

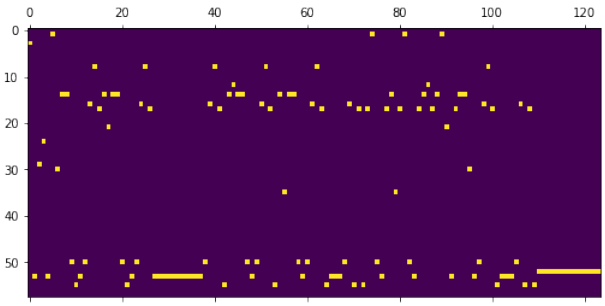


Figura B.2: Representação grafica de um Dummy

Bibliografia

[1] Word embeddings
https://www.tensorflow.org/text/guide/word$_e$mbeddings

[2] Zheng Xu, Sheng Wang, Feiyun Zhu, and Junzhou Huang
Seq2seq Fingerprint: An Unsupervised Deep Molecular Embedding for Drug Discovery BCB'17, Aug 2017, Boston, Massachusetts USA

[3] Issa Annamoradnejad, Gohar Zoghi (2021). ColBERT: Using BERT Sentence Embedding for Humor Detection.

[4] Na Pang1, Li Qian, Weimin Lyu,in-Dong Yang (2019). Transfer Learning for Scientific Data Chain Extraction in Small Chemical Corpus with joint BERT-CRF Model.

[5] James Briggs (2020). TensorFlow and Transformers.
https://towardsdatascience.com/tensorflow-and-transformers-df6fceaf57cc
Accessed 10 of May 2021.

[6] sameerpixelbot (2020). BERT Embeddings with TensorFlow 2.0.
https://www.kaggle.com/sameerpixelbot/bert-embeddings-with-tensorflow-2-0-example
Accessed 10 of May 2021.

[7] Dharti Dhami (2020). Understanding BERT — Word Embeddings.

https://medium.com/@dhartidhami/understanding-bert-word-embeddings-7dc4d2ea54ca
Accessed 10 of May 2021.

[8] Trishala Neeraj (2020). Feature-based Approach with BERT.
https://trishalaneeraj.github.io/2020-04-04/feature-based-approach-with-bert
Accessed 10 of May 2021.

[9] fchollet (2020). Transfer learning fine-tuning.
https://keras.io/guides/transfer$_l$earning/
$Accessed 10 of May 2021$.

[10] Craig A. James (2016). OpenSMILES specification.
http://opensmiles.org/opensmiles.html
Accessed 10 of May 2021.

[11] Samo Turk, Sabrina Jaeger, Simone Fulle. mol2vec Documentation - Release 0.1
https://mol2vec.readthedocs.io/en/latest/
Accessed 10 of May 2021.

[12] RDKit Cookbook
https://www.rdkit.org/docs/Cookbook.html
Accessed 16 of May 2021.

[13] Samo Turk - mol2vec
https://github.com/samoturk/mol2vec
Accessed 20 of May 2021.

[14] Shion Honda, Shoi Shi, Hiroki R. Ueda. (2019). SMILES Transformer: Pre-trained Molecular Fingerprint for Low Data Drug Discovery. Available at: https://arxiv.org/abs/1911.04738 [Accessed 7 of June 2021]

[15] Issa Annamoradnejad, Gohar Zoghi (2021). ColBERT: Using BERT Sentence Embedding for Humor Detection.