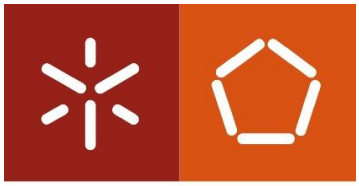

Universidade do Minho

Mestrado Integrado de Engenharia Informática

Aprendizagem Automática 2 (2ºSemestre - 2020/21)

Word embeddings com proteínas aplicado a classificação de enzimas

A85400 Nuno Azevedo Alves da Cunha

A85919 Pedro Dias Parente

Braga

6 de junho de 2021

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Este trabalho consiste no uso de técnicas de *Natural Language Processing* no contexto da classificação da função de proteínas através das suas sequências de aminoácidos.

A meta do trabalho foi tentar a partir da sequência de aminoácidos de enzimas efectuar a classificação das mesmas de acordo com o seu *E.C Number* usando word embeddings com trímeros de aminoacidos

Foram usadas diversas representações para as sequências de aminoácidos bem como modelos *DNN*, *CNN*, *RNN* e de Machine Learning tradicional para tentar atingir os resultados pretendidos.

Como apoio ao desenvolvimento do trabalho foram consultados diversos artigos científicos enunciados na secção de bibliografia.

Capítulo 2

Contextualização

Para este trabalho foi fornecido um dataset ("*ecpred_uniprot_uniref_90*") com 175806 linhas e 8 colunas. Cada linha do dataset corresponde a uma enzima, no entanto, no ambito deste trabalho apenas são releveantes 2 colunas:

- **sequence** - A sequência de aminoácidos correspondente à enzima
- **ec_number** - O Enzyme Comission Number

O Enzime comission Number é um sistema numérico de classificação de enzimas de acordo com as reações que estas catalizam. É composto por 4 niveis que classificam essas reações de forma cada vez mais especifica, o trabalho incidiu sobre a classificação deste número a vários nivéis.

A sequencia de aminoácidos é apenas uma string com as letras que representam cada aminoácido. Sobre esta sequencia é depois aplicada uma técnica de divisão em trímeros de aminoácidos de modo a poder aplicar os embeddings pré treinados do *Protvec*

Capítulo 3

Descrição do Trabalho e Análise de Resultados

3.1 Pre-processamento dos dados

Uma das grandes preocupações deste trabalho foi, pelo menos inicialmente, a transformação do dataset de modo a poderem ser aplicadas técnicas de Processamento de Linguagem Natural.

3.1.1 Sequência de aminoácidos para trímeros

De modo a aplicar técnicas de *NLP* é necessário encontrar semelhanças entre ambos os problemas. Neste caso isto passa por considerar a sequência de aminoácidos uma "frase"e os trímeros como "palavras". Neste sentido a primeira fase do pré-processamento passou pela separação da sequência de aminoàcidos em 3 listas de trimeros não *overlapping*.

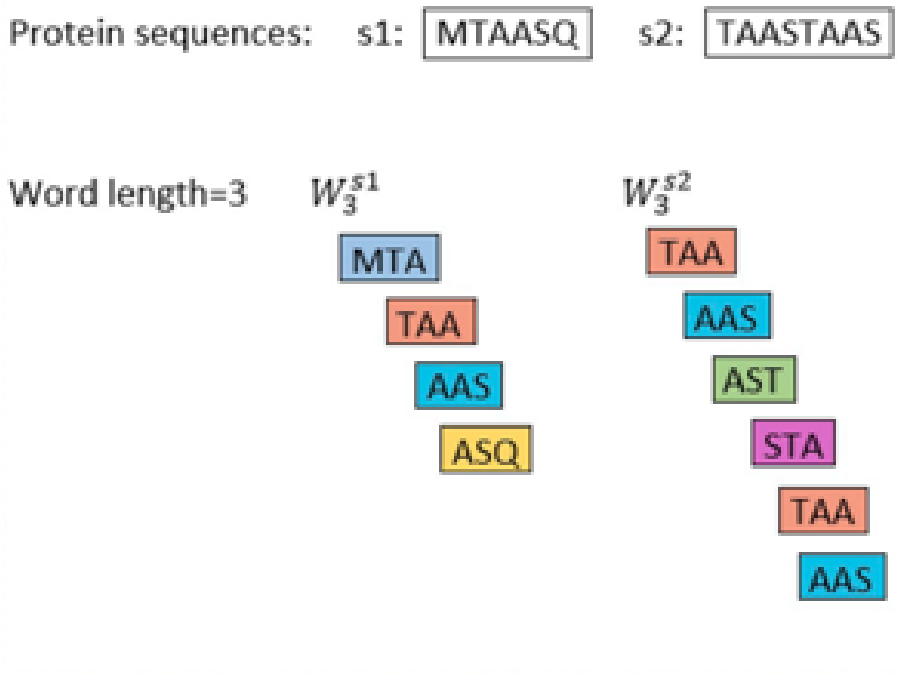

Figura 3.1: Separação da sequência em trímeros

3.1.2 Carregamento do embeddings pré-treinados Protvec

Como já exite algum trabalho feito em natural language processing o grupo usou um repositório de embeddings já treinados na base de dados de proteínas Swiss-prot. Estes embeddings fornecem um vocabulário de 9048 trímeros e os vetores de embeddings a eles associados que é depois aplicado nas representações de sequências do grupo

3.1.3 Representação de sequência num vetor de 100 dimensões

Baseado no artigo *"Protvec - Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics"* o grupo desenvolveu uma representação para a sequência de aminoácidos que consistem em somar todos os vetores associados aos trimeros da mesma num só obtendo assim como outuput um único vetor de dimensão 100.

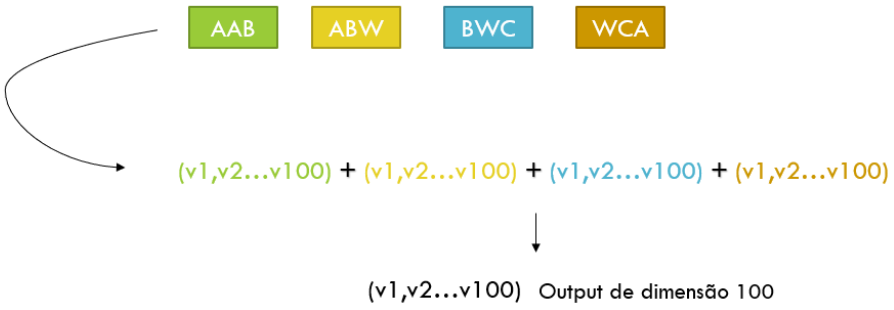

Figura 3.2: Representação de dimensão 100

3.1.4 Representação de sequência numa Matriz com padding

De acordo artigo *"SDN2GO: An Integrated Deep Learning Model for Protein Function Prediction"* o grupo implementou ainda uma representação em que os trimeros das sequência são subtituidos pelo seu vetor de embedding respetivo e depois é feito um padding com vetores 0 de 100 dimensões até a representação ser uma matriz de tamanho Max Length x 100 (Só são consideradas proteínas de tamanho Max Length ou inferior).

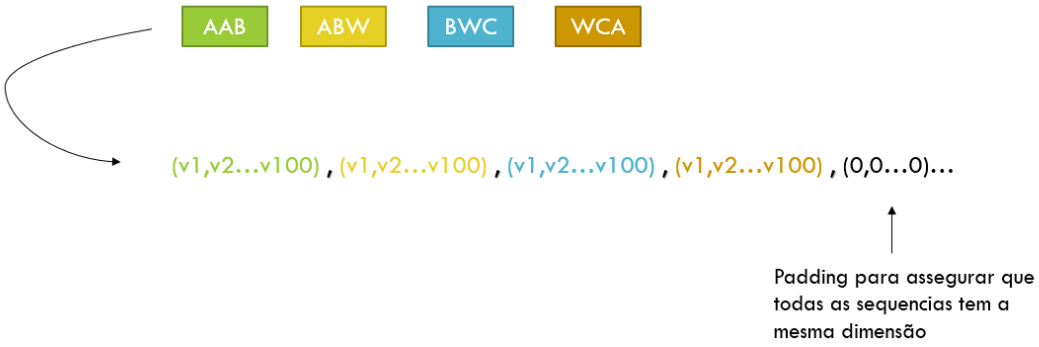

Figura 3.3: Representação em matriz

3.1.5 Representação de sequência com matriz de Vocabulário

Esta representação baseada no artigo *"TNFPred: identifying tumor necrosis factors using hybrid features based on word embeddings"* consiste numa matriz onde cada posição corresponde a uma palavra do vocabulário (ou seja um trímero) e o valor nessa posição é o vetor correspondente a esse mesmo trimero multiplicado pelo numero de ocorrências do mesmo na sequência.

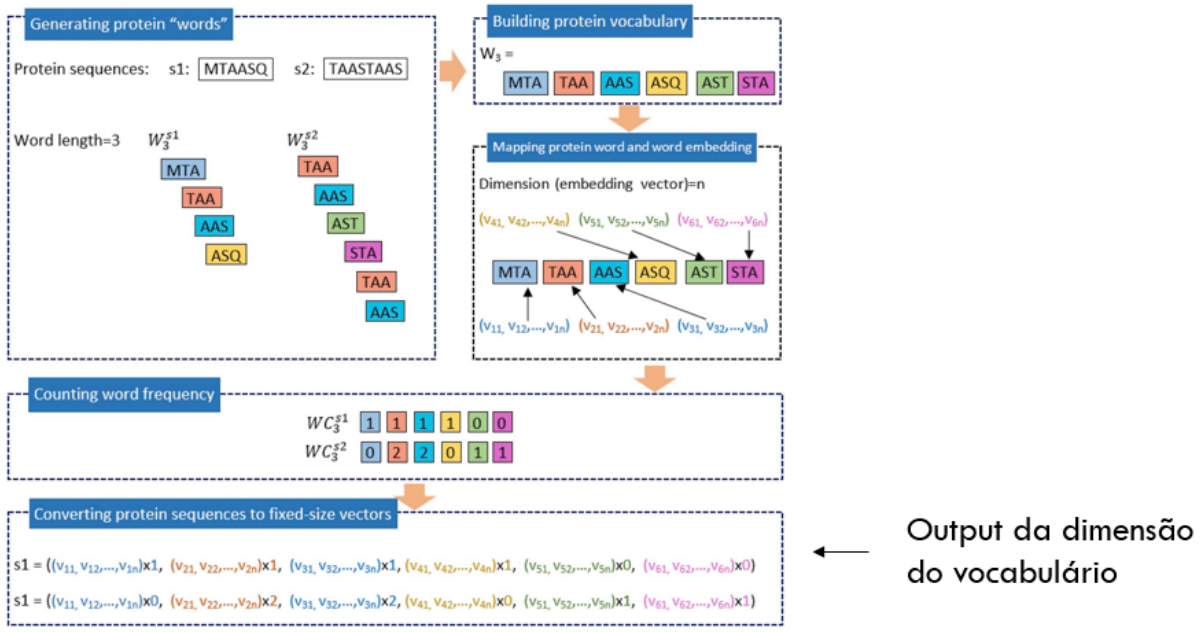

Figura 3.4: Representação com matriz de vocabulário

3.2 Desenvolvimento dos Modelos

3.2.1 Modelo RNN - LSTM

Inicialmente o grupo começou por desenvolver um rede neuronal recorrente, uma vez que estavam a ser usadas técnicas de NLP esta parecia a escolha mais natural porque normalmente apresenta muito bons resultados neste campo. inicialmente baseamos o nosso modelo no descrito no artigo *"Identifying antimicrobial peptides using word embedding with deep recurrent neural networks "* que recomendava 2 camadas GRU bidirecionais com altos valores de dropout. Isto acabou por produzir no entanto um modelo bastante underfitted para o nosso caso de estudo.

De modo a atacar este problema o grupo criou novos modelos substituindo as camdas GRU por camadas LSTM, baixando os valores de dropout e aumentando o número de neurónios em cada camada. Depois de ajustar estes valores para encontrar um bom balanço entre over e underfit o grupo chegou ao seguinte modelo.

```
model = models.Sequential()
model.add(Input(shape=(698, 100), dtype='float32', name='main_input'))
model.add(Bidirectional(LSTM(64, return_sequences=True)))
model.add(Dropout(0.3))
model.add(Bidirectional(LSTM(128)))
model.add(Dropout(0.5))
model.add(Dense(len(y[0]), activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['acc'])
```

Figura 3.5: "Modelo LSTM"

Para o treino e teste deste modelo foram usados os dados com as sequências na representação em forma de matriz com padding, no entanto esta representação produz matrizes massivas para cada sequência, apenas conseguimos então um um subset das 175000 linhas do dataset, cerca de 25000 exemplos ecolhidos aleatóriamente, o que ja produz um ficheiro com cerca de 11GB.

3.2.2 Modelo CNN

De modo a capitalizar da melhor forma possivel nas representações criadas com os embeddings foram também seguidos os modelos de cada um dos artigos de onde foram inspiradas essas representações. Neste caso o artigo *"SDN2GO: An Integrated Deep Learning Model for Protein Function Prediction"* utilizou uma rede convolucional para aproveitar a sua representação de matriz com padding.

A rede convolucional descrita no artigo era composta por 3 camadas Convolucionais 1D com activação relu e uma camada de MaxPooling associada a cada uma. Esta rede acabou por se provar satisfatória para o problema em questão, tendo o grupo que fazer apenas alguns ajustes no que toca a unidades em cada camada e a adição de um pequeno dropout para resolver um ligeiro overfiting. Acabando assim com o seguinte modelo:

```
model = models.Sequential()
model.add(Input(shape=(698, 100), dtype='float32', name='main_input'))
model.add(Conv1D(64, 3))
model.add(MaxPooling1D(3, 2))
model.add(Dropout(0.1))
model.add(Conv1D(128, 3))
model.add(MaxPooling1D(3, 2))
model.add(Dropout(0.2))
model.add(Conv1D(128, 3))
model.add(MaxPooling1D(3, 2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(len(y[0]), activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['acc'])
```

Figura 3.6: "Modelo CNN"

Mais uma vez o tamanho da representação impede a ultilização da totalidade do dataset para o treino deste modelo.

3.2.3 Modelo SVM/DNN

De acordo com o artigo onde foi inpirada a representação num único vetor de 100 dimensões o grupo começou por criar um modelo de machine learning tradicional com Support Vector Machines, no entanto este nao apresentou bons resultados apesar de, ao contrário das representações enunciadas anteriormente, ser possivel usar todo o dataset para o treino do modelo. Assim procurou-se desenvolver outro modelo para tirar vantagem desta representação a uma dimensão, neste caso um modelo DNN. Tentando encontrar um bom balanço entre complexidade a complexidade do modelo e overfitting o grupo foi optimizando os paramentros deste modelo até chegar ao seguinte modelo:

```
model = models.Sequential()
model.add(Input(shape=(100,), dtype='float32', name='main_input'))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.05))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(len(y[0]), activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['acc'])
```

Figura 3.7: "Modelo DNN"

Uma vez que a representação vectorial é mais reduzida em termos de espaço foi possivel treinar e testar este modelo de acordo com todo o dataset

3.3 Resultados

Os modelos foram em grande parte testados num servidor da Universidade do Minho uma vez que (principalmente os modelos RNN) poderiam ser bastante demorados num computador pessoal.

3.3.1 Modelo RNN

O modelo RNN mostrou, ao contrário do esperado para um tecnica baseada em NLP, um performance pouco satisfatória. Com um sampling de 25000 linhas do dataset este modelo mostrou uma PECC de cerca 55% para o primeiro digito do E.C Number. Isto é também justificável pelo modelo não poder ser treinado com todo o dataset, devido a dimensao do mesmo e das representações das sequências em matriz.

3.3.2 Modelo CNN

O modelo CNN portou-se ligeiramente melhor sendo capaz de classificar corretamente cerca de 60% dos exemplos de teste para o primeiro digito do E.C Number. Mais uma vez, fora problemas de under/overfitting que o grupo tentou reduzir ao máximo isto pode dever-se à quantidade reduzida de dados que é possivel usar com o tipo de respresentação usada neste modelo.

3.3.3 Modelo DNN

Dos modelos criados o modelo DNN foi o que apresentou resultados mais satisfatórios, com uma taxa de acerto superior a 70%, e tendo um bom balanço entre over e under fitting. Este modelo ao contrário dos referidos anteriormente teve a grande vantagem de poder ser treinado e testado com todo o dataset, dada a dimensão mais reduzida da representação vetorial. Foi também testada a classificação com vários niveis do E.C Number sendo que o modelo não mostrou uma queda muito significativa da performance com o aumento da especificidade do E.C number, embora tenha baixado um pouco a sua accuracy com o aumento do número de digitos a considerar.

Capítulo 4

Conclusão

Em conclusão, julgamos que face ao problema exposto o fomos capazes de aplicar as técnicas recomendadas pela orientadora para a aplicação de word embeddings na classificação de proteínas, procurando varias alternativas de aproximação ao trabalho de acordo com diversos artigos ciêntificos. Apesar disto os modelos implementados são ainda pouco fiaveis podendo isto querer dizer que as representações escolhidas pelo grupo não seriam as mais indicadas para esta problemática.

Durante a execução deste projeto, as nossas maiores dificuldades consistiram na ambientação ao problema para o qual tinhamos poucas bases do ponto de vista biológico, bem como na gestão das representações das sequencias de proteinas com word embeddings uma vez que estas produziam quantidades de dados muito grandes e o pré-processamento era bastante demorado.

Em futuras iterações do trabalho o grupo considera que seria vantajoso uma expansão do hardware ou uso de outras técnicas para poder ter as representações em matrizes com todo o dataset de modo a obter melhores resultados nos modelos CNN e LSTM. Era também interessante explorar melhor a representação com o vocabulário e wordcount que o grupo não foi capaz de testar de forma efetiva dada a sua dimensão.

Por fim gostaríamos de deixar um agradecimento à orientadora, Ana Marta Sequeira, pela grande disponilidade e recursos que forneceu ao grupo para que pudéssemos produzir o melhor trabalho possível.

Capítulo 5

Referências

5.1 Referências Eletrónicas

Protvec - Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. Disponíel em:
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0141287

SDN2GO: An Integrated Deep Learning Model for Protein Function Prediction. Disponíel em:
https://www.frontiersin.org/articles/10.3389/fbioe.2020.00391/full

TNFPred: identifying tumor necrosis factors using hybrid features based on word embeddings. Disponível em:
https://pubmed.ncbi.nlm.nih.gov/33087125/

Identifying antimicrobial peptides using word embedding with deep recurrent neural networks . Disponível em:
https://academic.oup.com/bioinformatics/article/35/12/2009/5172364