Departamento de Ciência da Computaçao - UFMG

Trabalho Prático I - Aprendizado Profundo para Processamento de Linguagem Natural

Thiago Malta Coutinho
thiagomaltac@gmail.com

October 6, 2019

1 Introdução

O objetivo desse trabalho prático é estudar o funcionamento do modelo de linguagem **Word2Vec** e explorar como os métodos **CBOW** e **Skip-gram** se comportam com a variação do tamanho da janela e do corpus. Para avaliar os resultados a similaridade de cossenos foi utilizada.

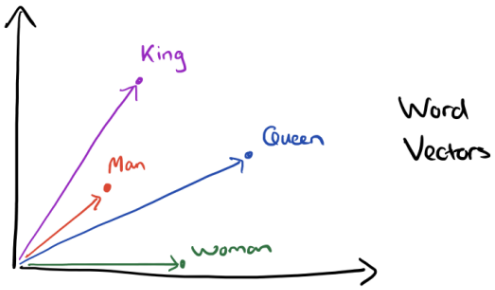

Figure 1.1: Representação das palavras como vetores.

1.1 Word2Vec

O modelo de linguagem *Word2Vec* trouxe grandes avanços em relação aos modelos de linguagem frequentistas, gerando representações das palavras que podem servir de entrada para outros modelos de aprendizado ou produzir resultados a partir da distância do cosseno. Ele utiliza uma rede neural com uma camada de neurônios sem função não-linear em sua saída, as palavras são as entradas e saídas da rede e são codificadas em formato *one-hot*; caso exista mais de uma palavra na entrada ou na saída, os vetores *one-hot* são concatenados. A rede projeta os vetores codificados em uma dimensão de tamanho igual

número de neurônios e otimiza os pesos da projeção com o objetivo de minimizar a função de custo presente na saída. A função de custo utilizada geralmente é a log-verossimilhança, atualizada com o algoritmo *Backpropagation*, e as probabilidades de cada classe(palavra) são estimadas utilizando a função *Softmax* logo após a saída da camada escondida.tempo Uma característica importante do algoritmo é a modelagem do contexto em que as palavras aparecem, podendo ser feita de duas formas: utilizando o *CBOW* ou *Skip-gram*, técnicas que serão apresentados nas Seções 1.1.1 e 1.1.2.

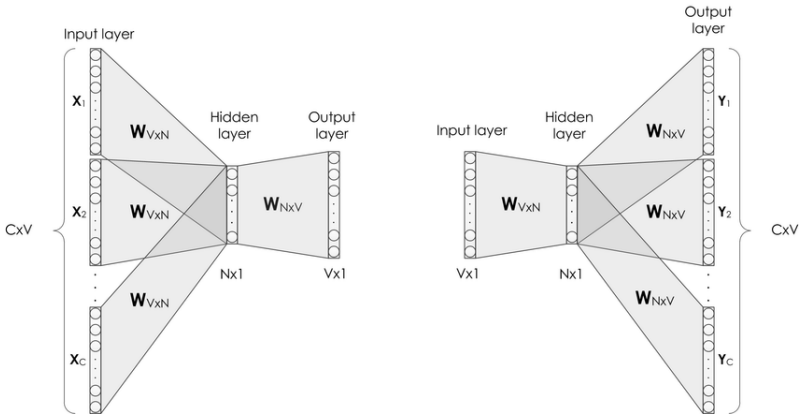

Figure 1.2: Estrutura do *Word2Vec* utilizando o *CBOW*(esquerda) e *Skip-gram*(direita).

1.1.1 CBOW

A abordagem *Continuous bag of words*(CBOW) é uma das técnicas utilizadas para modelar contexto. Nessa técnica, dado uma janela de contexto, a palavra central é a palavra a ser predita e as palavras restantes constituem o contexto linguístico atual.

1.1.2 Skip-gram

O Skip-gram funciona de forma análoga ao CBOW, dado uma janela de contexto a palavra central é a entrada e o restante das palavras são a saída da rede.

1.1.3 Similaridade de Cossenos

Os vetores gerados pelo modelo podem ser normalizados para possuírem norma unitária, pertencendo à casca de uma hiper-esfera n-dimensional, onde n é o tamanho da camada escondida. Para comparar vetores, a distância euclidiana se mostra pouco eficaz e a Similaridade de Cossenos se mostra mais apropriada. A Similaridade mede o quão dois vetores estão próximos, sua fórmula é apresentada a seguir:

$$cos_{similarity} = \frac{\sum (a-b)^2}{\sqrt{\sum a^2} \cdot \sqrt{\sum b^2}} \tag{1.1}$$

Onde `a` e `b` são vetores.

2 Experimento

A implementação utilizada para os experimento é o algoritmo desenvolvido pelo o Google em linguagem C, disponível no link https://code.google.com/archive/p/word2vec/. Tam-

bém foram desenvolvidas funções auxiliares em linguagem `python` e `shell` para processamento dos dados, treinamento dos modelos e processamento dos resultados.
A metodologia consiste em treinar um modelo de linguagem para cada conjunto de hiper-parâmetros especificados na Seção 2.3. A avaliação de desempenho dos modelos é feita utilizando o arquivo `word-analogies.txt` para gerar analogias com cada modelo treinado e comparar a palavra predita com a analogia correta, a comparação é feita com a Similaridade de Cossenos. O resultado final é a média de Similaridade.
Foi necessário fazer modificações nos arquivos `word-analogy.c` e `distance.c` para que as entradas fossem lidas de arquivos gerados por códigos intermediários em linguagem `python` e as saídas fossem formatadas de maneira a facilitar o experimento.

2.1 Corpus

O conjunto de dados de entrada do treinamento foi o *text8*, disponível no link https://mattmahoney.net/dc/text8.zip. O Corpus tem tamanho de 95.4MiB e contém 17.005.207 palavras, não possui pontuação e com algumas normalizações de texto aplicadas.

2.2 Pré-processamento

O pré-processamento dos dados foi feito com o auxílio da biblioteca `gensim`, em linguagem `python`. Palavras de tamanho menor que 3 foram retiradas com a função `strip_short`, as *stop-words* foram removidas chamando a função `remove_stopwords` e por fim os números entre 0 e 9(*one, two, three...*) escritos em extenso foram removidos.

2.3 Hiper-parâmetros

Os hiper-parâmetros avaliados nesse experimento foram a janela de contexto, o tamanho do dataset e se o CBOW ou o skip-gram seriam utilizados no algoritmo. Os tamanhos de dataset foram escolhidos em porcentagens do tamanho total, as porcentagens escolhidas foram **25%**, **50%**, **75%** e **100%**. Para selecionar as palavras que constituem os datasets filtrados foram retiradas as primeiras palavras do conjunto inicial de dados até que o número de palavras restantes correspondesse ao tamanho final desejado.

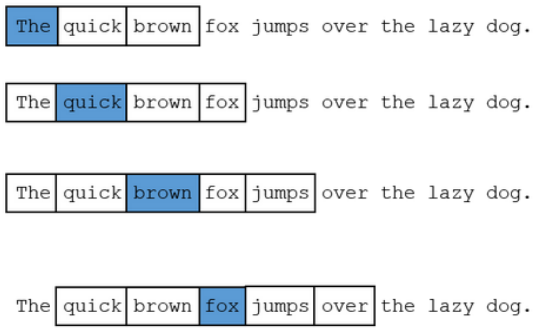

Figure 2.1: Janela de contexto se movimentando com as iterações do algoritmo.

O tamanho da janela de contexto foi variado entre o conjunto de valores **5**, **10**, **15** e **20**. O código do `Word2Vec` também possui funções de *negative sampling* e *hierarchical softmax*. Esses parâmetros não foram variados e o seu funcionamento não será abordado nesse trabalho. Os dois parâmetros assim como os demais foram mantidos os mesmos que vem de padrão com o código, com exceção dos parâmetros de tamanho da janela de contexto e se o algoritmo deve usar o CBOW ou o skip-gram.

3 Resultados

O conjunto de hiper-parâmetros apresentados na Seção 2.3 define 32 modelos de linguagem distintos. Os modelos foram treinados e avaliados de acordo com a distância do cosseno. Os resultados obtidos estão na Figura 3:

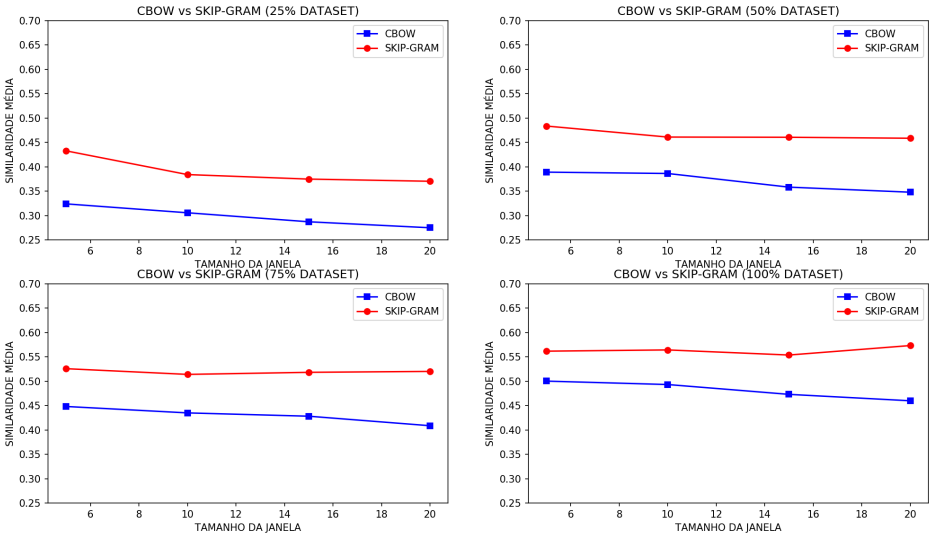

Figure 3.1: Resultados obtidos para o conjunto de 32 modelos `Word2Vec`.

A distância do cosseno é uma medida de similaridade e quanto maior seu valor, mais similar dois vetores são.
A partir da Figura 3 é possível visualizar que a abordagem usando o `skip-gram` forneceu, em média, respostas mais similares ao resultado correto em relação ao `CBOW`. Também observa-se que aumentar o tamanho do conjunto de dados melhorou o desempenho dos modelos, independentemente do tamanho da janela escolhida.

4 Conclusões

Nesse trabalho foi possível estudar o modelo de linguagem *Word2Vec* e entender melhor seu funcionamento. O aluno se familiarizou com a implementação fornecida pelo professor e apresentou uma rotina de avaliação do algoritmo ao variar os parâmetros especificados. Os resultados obtidos foram coerentes e esperados, o modelo `skip-gram` geralmente se comporta melhor que o `CBOW` em datasets pequenos como o utilizado. A melhora nos resultados com o aumento do conjunto de treinamento também é coerente, pois existe mais contexto para os modelos aprenderem e generalizar melhor.

5 Referências

1 - Notas de Aula do Professor Adriano Veloso, disciplina de Aprendizado Profundo para Processamento de Linguagem Natural.
2 - https://israelg99.github.io/2017-03-23-Word2Vec-Explained/