Análise do uso de NBOW para Classificação de Artigos Científicos

Vinícius Almeida dos Santos[1], **Guilherme Gustavo Gohr**[1], **Rafael de Santiago**[1]

[1] Laboratório de Inteligência Aplicada (LIA) – Universidade do Vale do Itajaí (UNIVALI) – Postal box 360 – 88.302-202 – Itajaí – SC – Brasil

viniciusas@edu.univali.br, guilhermegustavogohr@gmail.com, rsantiago@univali.br

Abstract. *The automatic document indexing is important because it is a hard and demanding time task for a human. For scientific papers, to automate this task is required due to the growing number of published papers. So, the research for automatic methods for paper indexing is relevant for the community in general. This paper presents an analysis of NBOW and TF-IDF for the task of indexing English papers from Computer Science area. Were compared different pre-trained word embedding sources, and the best results were obtained with a general precision of 68.18% using the word embedding LexVec-CC with TF-IDF. In particular, the most generic categories presented the worst classification.*

Resumo. *A automatização da indexação de documentos é importante, afinal é uma tarefa árdua e demanda muito tempo de um analista humano. Para artigos científicos, a automatização se torna fundamental dado a crescente publicação de artigos. Deste modo, o estudo de métodos automatizados de catalogação de artigos é relevante para a comunidade em geral. Este artigo apresenta uma análise do uso de NBOW com TF-IDF para a tarefa de catalogação de artigos em inglês da área de Ciência da Computação. Comparou-se várias fontes de word embedding pré-treinadas, e os melhores resultados foram obtidos com uma precisão geral de 68.18% utilizando LexVec-CC com TF-IDF, sendo que categorias mais genéricas apresentaram uma precisão pior na classificação.*

1. Introdução

A indexação de documentos é uma prática comum em diversos contextos, atribuindo os documentos aos grupos que melhor os descrevem. Essa atividade é realizada cada vez mais por conta do aumento de publicações e literatura técnico-científica. Consequentemente, aumenta a necessidade de recursos humanos e físicos alocados com esse propósito [Silva e Fujita 2004].

De acordo com [Lancaster 2004], a principal função da indexação de documentos é possibilitar a fácil inclusão em uma base de dados, sendo ela física ou eletrônica. A pesquisa de [Cunha 1999] indica que uma biblioteca digital bem catalogada/indexada facilita a utilização pelo usuário final, contanto que utilize esses dados para trazer informações mais relevantes ao usuário.

O trabalho de [Silva e Fujita 2004] levanta quatro aspectos que um leitor indexador pode estar sujeito: (*i*) tempo restrito para realizar a atividade; (*ii*) compreensão do texto apenas com o propósito de indexá-lo; (*iii*) a compreensão seguir a escrita de um

resumo; (*iv*) o leitor trabalhar em um contexto muito pequeno de assuntos e a repetição causar processamento automático. Um indexador humano dificilmente irá ler cada documento inteiro [Lancaster 2004], por isso métodos automáticos de indexação se tornam mais atrativos.

Existem vários métodos na literatura que abordam a problemática de classificação de elementos, como *Support Vector Machine* e Redes Recorrentes, que utilizam técnicas de aprendizado de máquina e mineração de dados. Nesse trabalho é avaliado um caso de uso do NBOW (*Neural Bag-Of-Words*) para classificação de textos, utilizando uma combinação de *word embedding* e redes neurais MLP (*Multi-Layer Perceptron*).

2. Definições

Para melhor compreender os resultados presentes neste trabalho, esta seção apresenta algumas definições. São tratados nesta seção o *Word Embedding*, NBOW e TF-IDF.

2.1. *Word Embedding*

Muitos métodos de processamento de linguagem natural lidam com cada palavra como entidade única, representando-a como uma posição de um vocabulário [Mikolov et al. 2013]. Entre os problemas do uso dessa prática estão a limitação de dados disponíveis (por exemplo, em reconhecimento de voz) e a desconsideração de similaridades entre palavras. O uso desses métodos deve ter dados envolvendo o máximo possível de palavras, pois em uma aplicação real, a precisão do sistema pode ser comprometida ao serem utilizadas palavras que estavam contidas no vocabulário, mas não na fase de aprendizado. Além disso, outro problema é relativo ao desempenho, pois uma palavra não aproveita processamento de palavras similares.

Uma alternativa a esses problemas é o uso dos *Word Embedding* (ou representação de palavras, vetores de palavra, *Word Vector*, *Word Features*) que consistem na representação de palavras em um espaço vetorial, de forma a criar propriedades específicas. Palavras similares estão relativamente próximas no espaço vetorial e acabam por demonstrar algumas particularidades [Mikolov et al. 2013].

Esse método se baseia em relações entre pares de *Word Embedding*, principalmente na relação de distância ou ângulo como método primário de avaliação da qualidade de um vocabulário. Um exemplo de afirmação que pode ser utilizada para validar é: "o rei está para rainha como um homem está para mulher", e representando em forma de equação $rei - rainha = homem - mulher$ [Pennington et al. 2014b].

Há várias implementações de vetores de palavra na literatura, mas foi optado GloVE (*Global Vectors*) [Pennington et al. 2014b] por ser relativamente recente, disponibilizar os vetores já treinados em [Pennington et al. 2014a] e ter melhores resultados conhecidos na literatura.

2.2. NBOW

O NBOW é um simples modelo de rede neural profunda com o propósito de uso em processamento de linguagem natural. Sendo um modelo que não releva a sintaxe, não é necessário realizar uma análise a cada palavra presente no texto, economizando tempo computacional. De acordo com o autor, esse modelo obtém resultados próximos ao

estado-da-arte em várias atividades a nível de frases e documentos com poucos minutos de treinamento [Iyyer et al. 2015].

Sendo X o conjunto de documentos em questão que serão classificados em k categorias, é aplicada uma função de composição g que irá unir todos os *word embeddings* $(v_w \mid w \in X)$ de forma a se tornar um único vetor z. [Iyyer et al. 2015]

A função de composição é utilizada para normalizar a entrada de dados na rede neural. Ela é descrita na Equação 1 [Iyyer et al. 2015].

$$z = g(w \in X) = \frac{1}{|X|} \sum_{w \in X} v_w \tag{1}$$

É possível utilizar o vetor resultante z para métodos de classificação textual. Testes realizados por [Iyyer et al. 2015] mostraram que a utilização da média como composição teve resultados melhores que simplesmente a soma, como utilizado no trabalho de [Kalchbrenner et al. 2014].

2.3. TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) [Salton e Buckley 1988] é um método não-supervisionado tradicional para peso de termos em textos [Timonen 2012]. Com o objetivo de identificar as palavras mais importantes nos documentos, TF-IDF é uma técnica que bonifica palavras presentes em muitos documentos e com uma baixa frequência em cada documento. Ela penaliza palavras pouco frequentes e presente em poucos documentos.

No somatório da equação *term frequency* $tf(w, X)$ (Equação 2, adaptado de [Timonen 2012]), é exemplificado uma busca sequencial no documento X pelo termo w, e somando 1 a cada vez que o termo y for igual ao termo w. O resultado do somatório é dividido pelo número total de palavras no documento $|X|$, obtendo assim o resultado da equação. Por outro ponto de vista, essa equação é o percentual do quanto que uma determinada palavra w está presente no documento X.

$$tf(w, X) = \frac{1}{|X|} \sum_{y \in X} \begin{cases} 1, & \text{se } y = w \\ 0, & \text{senão} \end{cases} \tag{2}$$

A equação *inverse document frequency* $idf(w)$ (Equação 3) realiza um somatório que soma 1 a todo documento X pertencente ao conjunto de todos os documentos D que contém a palavra w em seu conteúdo, divide esse somatório pela quantidade de documentos analisados $|D|$, formando o percentual de documentos que contém a palavra w. Depois é aplicada a operação de log na base 10, e colocado o resultado positivo. Uma alternativa para a adaptação dessa equação é inverter a divisão e remover o sinal negativo da frente da equação.

$$idf(w) = -\log_{10}\left(\frac{1}{|D|} \sum_{X \in D} \begin{cases} 1, & \text{se } w \in X \\ 0, & \text{senão} \end{cases}\right) \tag{3}$$

A fórmula final *term frequency-inverse document frequency* $tfidf(w, X)$ (Equação 4) é o produto entre $tf(w, X)$ e $idf(w)$, e resulta na importância da palavra w no documento X.

$$tfidf(w, X) = tf(w, X) \times idf(w) \tag{4}$$

3. Desenvolvimento

A implementação do modelo foi feita em C++, com a utilização da biblioteca OpenMP para paralelismo, paralelizando as operações de *backpropagation* e *feedforward* da rede neural. O desenvolvimento consistiu em: uma rede neural *feed-forward* com aprendizado supervisionado; funções para leitura das informações da base de dados; carregamento dos *word embedding* pré-treinados; cálculo do TF-IDF; e implementação do NBOW normal e com *dropout*.

3.1. NBOW com TF-IDF

Para melhoramento da qualidade do modelo NBOW, utilizou-se TF-IDF ainda no processo de composição do *word embedding* do documento. Uma complicação é como calcular a importância para um documento que não está listado nos documentos na fase de treinamento, torna-se então necessário utilizar uma equação geral para calcular a importância da palavra independente do documento em que está inserida.

Consideramos então I (Equação 5) o conjunto das importâncias de cada uma das palavras w inseridas no vocabulário V. Uma restrição desse método é que o vocabulário ficará restrito às palavras que estiverem em pelo menos um documento X no conjunto de documentos D que serão utilizados na fase de treinamento, sendo que se fossem buscadas palavras não presentes inicialmente, o resultado da importância seria 0.

$$I = \{I_w | w \in V\} \tag{5}$$

Para calcular uma importância global de uma palavra (Equação 6), sem depender de um documento, define-se uma média da importância $tfidf$ da palavra w em cada documento X do conjunto D presente na fase de treinamento.

$$I_w = \frac{1}{|D|} \sum_{X \in D} tfidf(w, X) \tag{6}$$

É obtida então a importância para cada palavra individualmente, para utilizá-las é necessário adaptar a função de composição do NBOW para comportar as importâncias das palavra. Nos testes preliminares, utilizar simplesmente o somatório ponderado das palavras divido pelo número de palavras $|X|$ no documento X não teve bons resultados, o que justifica a Equação 7, que obtém melhores taxas de acerto.

Onde é realizada a média ponderada dos *word embedding* v_w de cada palavra w com peso I_w. A Equação 7, baseada na Equação 1, também gera um *word embedding* resultante que representa a média do texto de entrada.

$$z = g(w \in X) = \frac{1}{\sum_{w \in X} I_w} \sum_{w \in X} I_w \times \upsilon_w \quad (7)$$

Tendo definida a função de composição com peso em cada *word embedding*, é possível então utilizar a saída z da equação 7 como entrada de uma rede neural supervisionada, mas qualquer método de *machine learning* já poderia resolver após esse passo.

3.1.1. Otimização

A fase de treinamento requer tempo computacional considerável, pois devem ser calculadas as médias de *word embedding* para todos os documentos a cada época. Sendo que documentos podem conter muitas palavras, é possível armazenar o vetor resultante, que serve como entrada da rede, em um momento anterior ao treinamento, exigindo mais memória, mas menos processamento. O problema do uso essa otimização é tornar inviável o uso da técnica de *dropout*, introduzida por [Iyyer et al. 2015], que de acordo com o autor apresenta melhoras na qualidade do treinamento.

4. Análise dos Resultados

A tarefa de processamento de linguagem natural a ser realizada, como comentado na seção 1, é a indexação/categorização de documentos, usualmente utilizada para divisão entre categorias das publicações de artigos em geral. Os documentos utilizados são artigos de computação baixados do site *Science Direct*. Foram obtidos 150 artigos de cada uma das 12 categorias, mostradas na Tabela 1.

Computer Science Applications	Computer Science (General)
Computer Graphics	Hardware
Artificial Intelligence	Human-Computer Interaction
Signal Processing	Computer Networks
Information Systems	Software
Computer Theory and Math	Computer Vision and Pattern Recognition

Tabela 1. Categorias dos documentos obtidos para a tarefa de classificação

A seguir são apresentados os detalhes dos *word embeddings* obtidos, dos experimentos realizados, e a análise da classificação.

4.1. *Word Embedding*

É possível obter vetores pré-treinados de *word embedding*, foram optadas pelas seguintes fontes de dados:

Identificador	Fonte dos dados	Tamanho do vetor	Autor
GloVe50	*Dataset* padrão disponível com o código fonte	50	[Pennington et al. 2014b]
GloVe300	*Common Crawl* pré-treinado disponível no site do GloVe (840B tokens)	300	[Pennington et al. 2014b]
Word2Vec	Baixado 'GoogleNews-vectors-negative300.bin' disponível online e convertido para texto	300	[Mikolov et al. 2013]
LexVec-EW	Diponível no GitHub, *Common Crawl* (58B tokens)	300	[Salle et al. 2016a]
LexVec-CC	Diponível no GitHub, *English Wikipédia 2015* + *NewsCrawl* (58B tokens)	300	[Salle et al. 2016a]

Tabela 2. Fontes dos vetores pré-treinados de *word embedding*

De acordo com [Pennington et al. 2014b], o modelo GloVe apresenta melhores resultados em comparação com muitas outras opções de *word embedding* no teste de analogia entre palavras, e também principalmente na precisão da relação sintática entre elas, inclusive em comparação com o popular método Word2Vec de [Mikolov et al. 2013].

O modelo LexVec de [Salle et al. 2016b, Salle et al. 2016a] é o estado da arte de modelos de contagem, sendo melhor que o estado da arte anterior em analogias sintáticas.

4.2. Experimentos

Para executar experimentos na tarefa de classificação dos documentos, foram utilizadas: 3000 épocas; *learning rate* de 0.4; tamanho da camada escondida 20; ativação sigmóide; com e sem a adição de TF-IDF na função de composição, conforme descrito na seção 3.1.

Para testes foram utilizados 30% dos 150 documentos obtidos, sendo 45 para testes e 105 para treinamento.

Na validação, foi retirada 1 palavra a cada 3 em cada um dos 45 documentos destinados para treinamento, resultando em 30% do texto de cada um dos documentos do treinamento para validação.

Os testes foram executados em um computador com Ubuntu 16.04, 8GB de RAM, SSD e um processador Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz com 4 núcleos de processamento.

No gráfico da Figura 1, é possível notar uma grande diferença de tempo ao comparar o tamanho do *word embedding* de 50 para 300, sendo uma diferença média do tempo de treinamento de 4 vezes, enquanto o tamanho do vetor é 6 vezes maior. Houve uma diferença pequena de tempo ao utilizar NBOW com TF-IDF, em alguns casos o tempo de treinamento foi até levemente menor, possivelmente pela restrição do vocabulário para somente as palavras que foram utilizadas no treinamento e cálculo das importâncias.

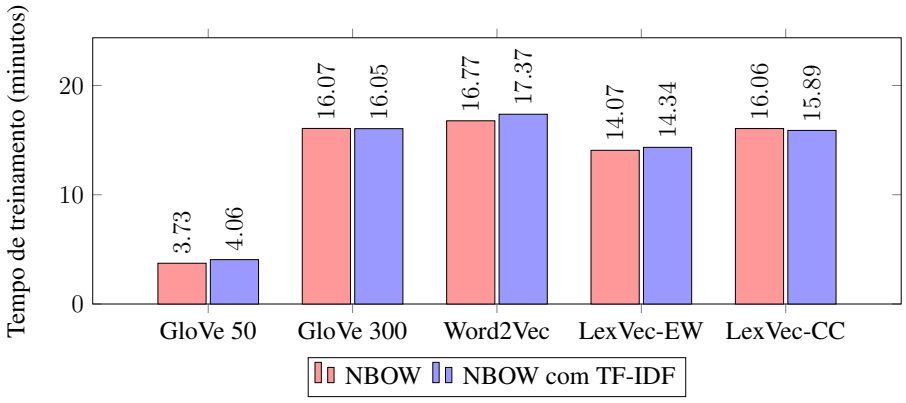

Figura 1. Tempo de treinamento para cada opção de *word embedding*

Na Figura 2, comparando a qualidade do uso de cada uma das fontes de *word embedding* com e sem o uso de TF-IDF, percebe-se uma melhora média de 8,4% na classificação nos *word embedding* de tamanho 300, e com tamanho 50 há uma melhora de 7,01%. Curiosamente, o GloVe50 com TF-IDF obteve uma taxa de acerto igual a do Word2Vec sem TF-IDF.

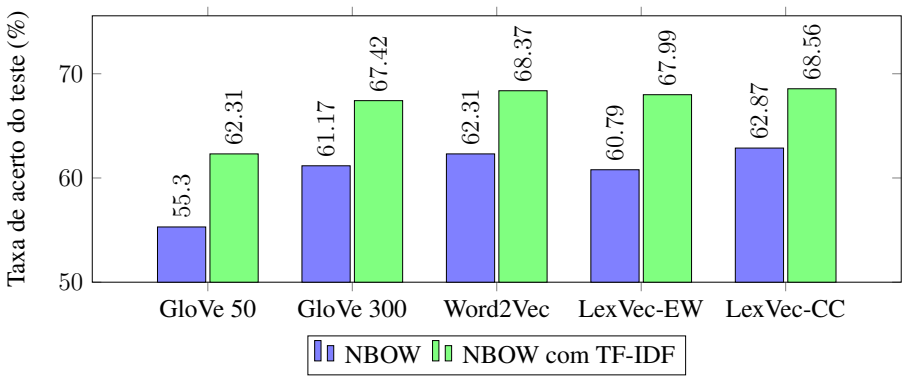

Figura 2. Qualidade máxima do treinamento para cada opção de *word embedding*

A taxa de acerto é calculada pela porcentagem de documentos que tiveram a maior categoria sendo igual à categoria correta, testando uma possível aplicação em um sistema real que iria categorizar os artigos de acordo com a predição mais provável fornecida pela rede.

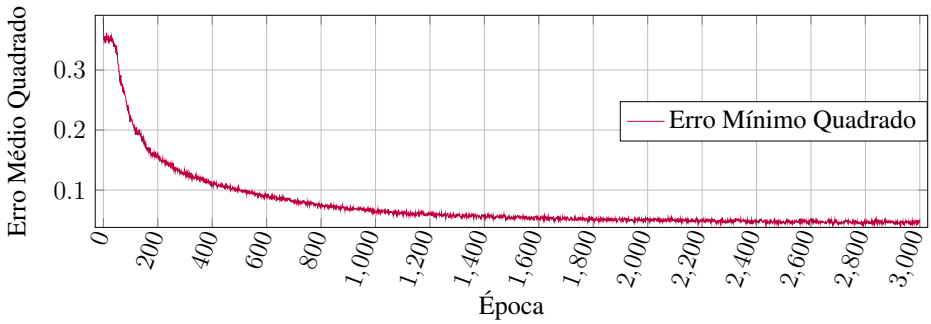

Figura 3. Erro da execução de NBOW com LexVec-CC com TF-IDF

Na Figura 3, é apresentado um gráfico onde o eixo vertical representa o erro médio quadrado e no eixo horizontal a época correspondente ao erro. É possível notar que o erro diminui consideravelmente até aproximadamente a época 1000, após isso houve pouca melhora. Essa pouca melhora representa uma saturação da rede, buscando melhores soluções em um ótimo local, mas sem conseguir melhoras que sejam relevantes.

Uma época consiste em: (*i*) embaralhar os documentos de forma a ficarem em uma ordem aleatória; e (*ii*) executar as operações de *backpropagation* para cada um dos documentos.

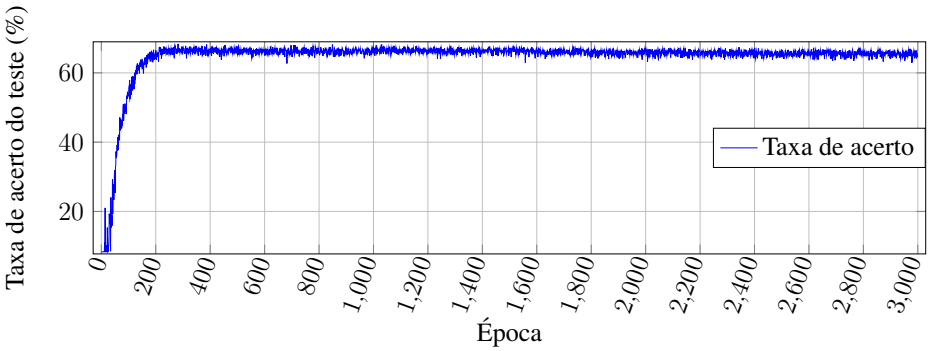

Figura 4. Taxa de acerto do teste de NBOW com LexVec-CC com TF-IDF

Os documentos de teste, que foram obtidos ao separar 30% dos artigos e não utilizá-los na fase de treinamento, comprovam se a rede está realmente acertando as predições, e as taxas de acerto indicam a quantia de documentos que seriam classificados corretamente em uma situação real. Na Figura 4, nota-se que próximo à época 400 a rede apresenta os melhores resultados do teste, identificando a região de soluções que contém os melhores resultados para classificar instâncias externas. Nota-se também que após isso, a taxa de acerto do teste reduz de forma constante, indicando que a rede está começando a decorar os dados do treinamento, deixando de generalizar a classificação.

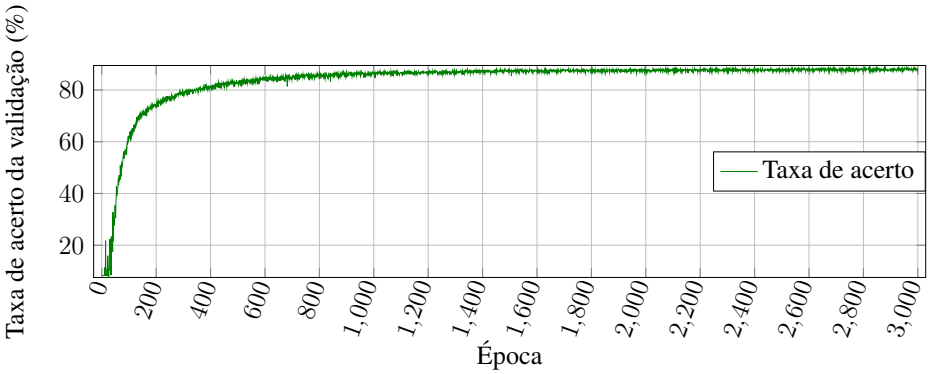

Figura 5. Taxa de acerto da validação de NBOW com LexVec-CC com TF-IDF

Os documentos de validação contém, como comentado na seção 4.2, uma palavra a cada três de cada um dos documentos de treinamento, tendo então 30% de cada documento. A cada época, é realizado um teste da taxa de acerto dessa validação. Há um

grande aumento na taxa de acerto dessa validação nas primeiras 200 épocas, mas nota-se que essa curva continua aumentando após muito tempo, indicando a rede deixando de generalizar os dados e começando a decorar as entradas do treinamento.

4.3. Resultados e Análises

Com base nos testes é perceptível que entre as 3000 épocas, os melhores resultados foram obtidos em menos que 400 épocas. Torna-se então, desnecessário rodar a rede por mais tempo, o que reduz consideravelmente o tempo de 16 minutos apresentados na Figura 1.

Categoria	**Acerto**	**Categoria**	**Acerto**
Computer Science Applications	52.27%	*Signal Processing*	65.90%
Computer Science (General)	***29.54%***	*Computer Networks*	79.54%
Computer Graphics	**90.90%**	*Information Systems*	68.18%
Hardware	70.45%	*Software*	77.27%
Artificial Intelligence	68.18%	*Computer Theory and Math*	59.09%
Human-Computer Interaction	68.18%	*Computer Vision and Pattern Recognition*	**93.18%**

Tabela 3. Taxa de acerto de cada categoria

Separando a taxa de acerto de cada categoria na Tabela 3, é possível ver a taxa de acerto que cada categoria teve individualmente. Percebe-se que *Computer Science (General)* teve a pior taxa de acerto, enquanto as melhores foram para *Computer Vision and Pattern Recognition* e *Computer Graphics*.

A partir da tabela 1, é possível notar que a rede realmente teve resultados ruins para o propósito de indexação, sendo que uma pessoa pode depender de uma boa indexação para encontrar o artigo que deseja.

Possivelmente, a categoria *Computer Science (General)* teve resultados ruins pois é uma categoria muito genérica, já as categorias que obtiveram os melhores resultados aparentemente tem sua média de *word embedding* bem diferente em comparação com as outras categorias.

5. Conclusão

Nesse artigo, analisamos o uso de vários *word embedding* para uso com o método NBOW para a tarefa de classificação de documentos. Essa classificação pode ajudar à eliminar a atividade humana e também aumentar a velocidade em que artigos científicos podem ser classificados.

O método TF-IDF colaborou com uma boa melhora na precisão de classificação, melhorando em 8% de taxa de acerto.

Como trabalhos futuros, sugere-se: experimentar (*i*) o uso de vários *word embedding* como entrada da rede; (*ii*) utilizar outras funções de importância de palavra; (*iii*) testar a melhora que uma camada softmax apresentaria na rede; (*iv*) indexar outras bases de dados; (*v*) substituir a rede neural por outro método de classificação; (*vi*) treinar *word embeddings* específicos para os artigos de computação.

6. Referências

Cunha, M. B. d. (1999). Desafios na construção de uma biblioteca digital. *Ciência da Informação*, 28:257 – 268.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188.

Lancaster, F. W. (2004). Indexação e resumos: teoria e prática.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. 2013.

Pennington, J., Socher, R., and Manning, C. D. (2014a). Global vectors for word representation. https://nlp.stanford.edu/projects/glove/. Acesso em: 2017-11-04.

Pennington, J., Socher, R., and Manning, C. D. (2014b). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Salle, A., Idiart, M., and Villavicencio, A. (2016a). Enhancing the lexvec distributed word representation model using positional contexts and external memory. *CoRR*, abs/1606.01283.

Salle, A., Idiart, M., and Villavicencio, A. (2016b). Matrix Factorization using Window Sampling and Negative Sampling for Improved Word Representations. *ArXiv e-prints*.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523.

Silva, M. d. R. d. S. and Fujita, M. S. L. (2004). A prática de indexação: análise da evolução de tendências teóricas e metodológicas. *Transinformação*, 16:133 – 161.

Timonen, M. (2012). Categorization of very short documents. *In-press KDIR'12*, pages 5–16.