Evaluation of Dimensionality Reduction and Truncation Techniques for Word Embeddings

Paulo Henrique Calado Aoun[1], **Andre C. A. Nascimento**[1], **Adenilton J. da Silva**[1]

[1] Departamento de Computação
Universidade Federal Rural de Pernambuco (UFRPE)
Recife – PE – Brasil

paulo_calado@live.com,{andre.camara,adenilton.silva}@ufrpe.br

Abstract. *The use of word embeddings is becoming very common in many Natural Language Processing tasks. Most of the time, these require computacional resources that can not be found in most part of the current mobile devices. In this work, we evaluate a combination of numeric truncation and dimensionality reduction strategies in order to obtain smaller vectorial representations without substancial losses in performance.*

Resumo. *O uso de word embeddings está se tornando muito comum em diversas tarefas de processamento de linguagem natural. Na maioria das vezes, eles exigem recursos computacionais que não podem ser encontrados na maior parte dos dispositivos móveis atuais. Neste trabalho, avaliamos uma combinação de estratégias de truncagem numérica e redução de dimensionalidade para obter representações vetoriais menores sem perdas substanciais no desempenho.*

1. Introdução

Em diversas tarefas de processamento de linguagem natural (PLN) a identificação do significado de uma palavra é de fundamental importância. Recentemente, a utilização de representações vetoriais, obtidos de forma não supervisionada, tem-se destacado [Levy and Goldberg 2014]. Uma vez que tal vetor seja obtido, é possível calcular a proximidade entre palavras e portanto inferir um significado ou contexto para elas.

No entanto, um fator impactante é o tamanho adotado na representação vetorial, dado que quanto maior o número de dimensões, maior o custo de treinamento. O mesmo pode se dizer do custo de memória e armazenamento, que também será maior. Tal aspecto é primordial em contextos em que recursos computacionais são ainda mais limitados, como, por exemplo, em dispositivos móveis, e.g. *smartphones* e *tablets*. Em [Ling et al. 2016], é demonstrado que 1 milhão de palavras representadas por vetores de 200 dimensões chegam a ocupar 1.6GB de memória em um sistema operacional de 64-bits.

Como [Ling et al. 2016] apresenta, a redução da precisão (para binário, 4, 6 e 8 bits) em representações treinadas previamente pode obter resultados similares aos obtidos com precisão de 64 bits em tarefas de similaridade de palavras e de *parsers* de dependência. Isso mostra que a redução da precisão, não necessariamente representa uma perda substancial no desempenho. Em [Raunak 2017], experimentos foram realizados para avaliar diferentes abordagens de redução de dimensionalidade, obtendo resultados

equivalentes mesmo quando o número de dimensões de vetores treinados previamente foi reduzido pela metade.

Entretanto, até o persente momento não foram realizados trabalhos que avaliem a combinação destas duas abordagens de compactação de informação. Este trabalho visa avaliar até que ponto a redução de dimensionalidade combinada a truncagem de valores afeta o desempenho de representações vetoriais de palavras em tarefas de similaridade léxica. Mais especificamente, é realizada a combinação de duas técnicas apresentadas em trabalhos anteriores [Raunak 2017, Ling et al. 2016], de diminuição da quantidade de bits (onde por padrão, são utilizados 64) para representação de uma dimensão do vetor [Ling et al. 2016] e de redução de dimensionalidade, utilizando o algoritmo *Post-Processing Algorithm* (PPA) em conjunto com o algoritmo *Principal component analysis* (PCA) [Raunak 2017].

Buscamos dessa forma, oferecer uma visão abrangente sobre os limites em que a perda de informação não inviabilize o uso de tais representações. Também apresentamos uma análise da redução do tamanho do arquivo resultante, consequentemente, diminuindo o espaço necessário para utilização do mesmo.

1.1. *Word Embeddings*

Word Embeddings são um tipo de representação de palavras com significados semelhantes e que possuem representações similares. Um dos principais benefícios do uso de vetores densos e de poucas dimensões, é que a maioria das redes neurais não performa de forma satisfatória com vetores de grandes dimensões. *Word embeddings* podem então ser definidos comouma classe de técnicas onde cada palavra é individualmente representada como um vetor de valores reais em um espaço vetorial predefinido [Goldberg 2017].

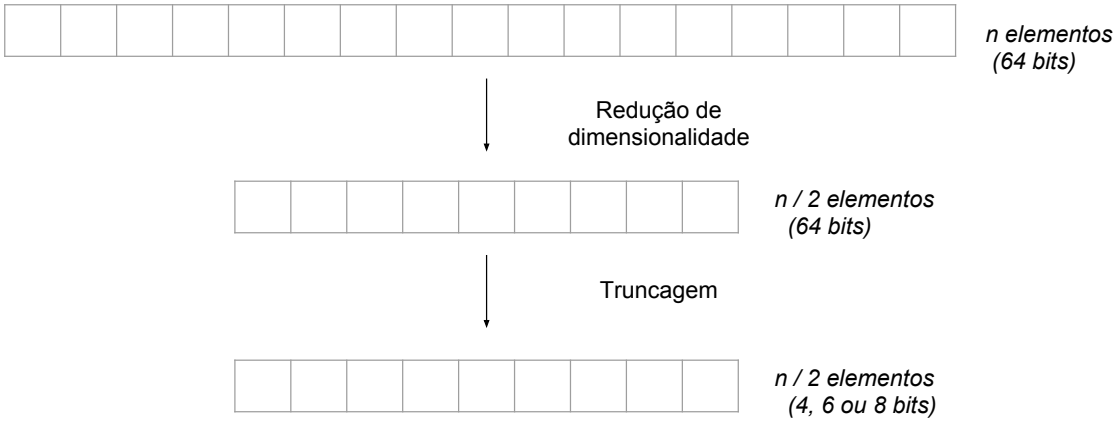
[Goldberg 2017] Também mostra que tais representações, possuem cerca de dezenas ou centenas de dimensões, em contraste as milhares ou às vezes, milhões de dimensões necessárias para representações como *one-hot encoding*. As representações distribuídas são aprendidas baseado na utilização das palavras. Isso faz com que as palavras de utilizadas de forma semelhante resultem em representações semelhantes, capturando naturalmente seu significado.

O restante deste trabalho está dividido em 4 seções. A Seção 2 descreve os métodos utilizados neste trabalho. A Seção 3 apresenta os experimentos. Os resultados obtidos são apresentados na Seção 4 e a Seção 5 é a conclusão.

2. Métodos

Nesta seção descrevemos as etapas adotadas para a redução de dimensionalidade e de precisão dos vetores de palavras (Figura 2). A primeira etapa consiste na aplicação dos algoritmos de redução de dimensionalidade PCA [Shlens 2014] e PPA [Mu et al. 2017], como proposto em [Raunak 2017]. Este método de redução de dimensionalidade foi escolhido pela sua eficiência em construir *word embeddings* com menores dimensões, e ainda manter desempenho similar ou até melhor (os autores demonstraram que em 7 dos 12 datasets utilizados no experimento, os *word embeddings* de 150 dimensões gerados a partir da aplicação do algoritmo, obtiveram resultados em média 2.74% melhores do que os *embeddings* de 300 dimensões).

Figura 1. Etapas de pré-processamento de *word embeddings*.



O passo seguinte é a aplicação dos métodos de truncagem propostos por [Ling et al. 2016], este método foi escolhido por ser possível reduzir significativamente o número de bits necessário por dimensões em modelos com até 200 dimensões. Os autores demonstraram que 8 bits por dimensão é o suficiente para representação dos valores e manter o desempenho em problemas de similaridade semântica, uma redução significativa em comparação aos 64 bits utilizados em grande parte dos sistemas atuais. Ambos os métodos[Raunak 2017, Ling et al. 2016] são aplicados sobre um conjunto de vetores treinados previamente.

Na seção 2.1, serão detalhados os métodos de redução de dimensionalidade considerados neste trabalho. A seção 2.2 apresenta o algoritmo de truncagem adotado nos experimentos.

2.1. Redução de dimensionalidade

O espaço na memória necessário para o armazenamento da representação vetorial das palavras (*word embeddings*) é diretamente proporcional a dimensionalidade do mesmo. Por este motivo, a redução da dimensionalidade dos vetores se torna essencial para a redução do tamanho do arquivo resultante. As abordagens descritas a seguir são as utilizadas no trabalho para tratar o problema da redução da dimensionalidade.

PCA

O algoritmo *Principal component analysis* (PCA) [Shlens 2014] permite reduzir a dimensionalidade de uma base dados através da combinação de atributos em novos atributos (i.e., componentes). Por fim, um subconjunto das componentes geradas pode ser utilizada como uma nova representação (reduzida) dos dados originais. Em [Raunak 2017], é adotado um método de redução de dimensionalidade de *embeddings* baseado em PCA, o qual descrevemos a seguir:

1. Organizar o dado em uma matriz $n \times m$, onde n é a dimensionalidade dos vetores de palavras e m é o número de palavras do dicionário;
2. Subtrair a média de cada tipo de medida.
3. Aplicar o algoritmo PCA sobre a matriz resultante, e selecionar os $n/2$ componentes principais.

PPA

O algoritmo de pós processamento (*Post Processing Algorithm*, PPA)[Mu et al. 2017] baseia-se em observações geométricas que mostram *word embeddings* obtidas após a subtração de parte de sua energia (vetor médio), de modo que a informação possa ser representada em um subespaço com menos dimensões. Este algoritmo parte da premissa de que todas as representações de palavras possuem o mesmo vetor comum u e possuem as mesmas direções dominantes D. Sendo assim, o vetor comum e as direções influenciam fortemente a representação de palavras da mesma forma, de modo que o algoritmo busca eliminá-los. O algoritmo PPA é apresentado no Algoritmo 1.

Algoritmo 1: Algoritmo PPA(X,D)

Data: Matriz de *Word Embedding* X, D
Result: Matriz *Word Embedding* Pós-processada X
Subtrair a média:
X=X-mean(X).
Computar componentes PCA:
u_i=PCA(X), onde i=1,2,...,d
Onde d=dimensionalidade de *embedding*
Eliminar maiores componentes: $\forall v$ em X
$v = v - \sum_{i=1}^{D}(u_i^T \times v)u_i$
Onde v é a representação de uma palavra

Raunak, 2017

Um outro algoritmo, proposto por [Raunak 2017] consiste em uma pequena modificação no algoritmo PPA. Esta abordagem adota a aplicação do algoritmo PCA em sequência, o que por sua vez leva a uma representação ainda mais compacta em termos de densidade do vetor resultante (Algoritmo 2).

O algoritmo baseia-se em três ideias principais: (i) o algoritmo PPA, comprovadamente resulta em melhores *embeddings*. (ii) quanto maior o pós-processamento, menor as dimensões dos *embeddings*, pois projetando de uma forma mais distante os vetores de palavras de suas direções dominantes, é possível gerar melhores *embeddings*. (iii) A extensão dos principais componentes explica o porque dos dados não serem tão bons como nos *embeddings* originais de 300 dimensões.

2.2. Redução do número de bits

A redução da precisão consiste na truncagem dos valores x em cada posição do vetor de representação de palavras, em uma representação utilizando n bits [Ling et al. 2016]. Por exemplo, para arredondar um valor x que fique dentro de um conjunto de valores [-*r*,*r*] onde *r* é dado por:

$$r = 2^{n-1} - 1. \tag{1}$$

Algoritmo 2: Algoritmo Raunak,2017

Data: Matriz *Word Embedding* X, Parâmetro Limiar D, Nova Dimensão N
Result: Matriz *Word Embedding* X com Dimensão reduzida N
Aplicar o Algoritmo de Pós-Processamento:
X = PPA(X, D)
Transformar X usando PCA:
X = PCA(X)
Aplicar o Algoritmo de Pós-Processamento:
X = PPA(X, D)

E a função de arredondamento pode então ser expressa como:

$$R(x,n) = \begin{cases} \lfloor x \rfloor, & se \quad x \leq \lfloor x \rfloor + \frac{\epsilon}{2} \\ \lfloor x \rfloor + \epsilon, \end{cases} \tag{2}$$

com ϵ sendo definido como:

$$\epsilon = 2^{1-n} r. \tag{3}$$

3. Experimentos

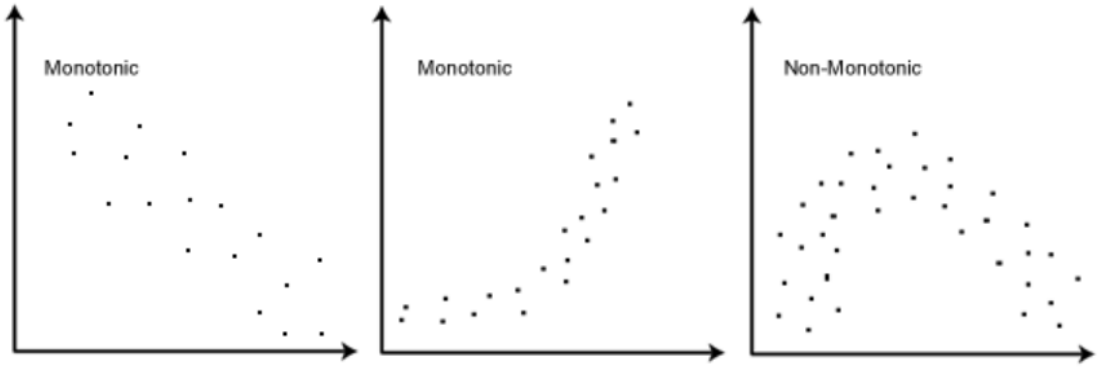
Os experimentos foram realizados no contexto de similaridade semântica de palavras, utilizando os benchmarks resumidos por [Faruqui and Dyer 2014]. Cada base de dados é composta por pares de palavras, cuja similaridade foi avaliada por humanos. Ao avaliar vetores de palavras, a similaridade entre as palavras é calculada pelo cosseno de suas representações vetoriais. Em seguida, calcula-se o coeficiente de correlação de Spearman [Well and Myers 2003] entre as classificações produzidas automaticamente e as anotadas manualmente. A correlação de Spearman mede a força e a direção entre duas variáveis com relação monotônica entre elas, ao invés de medir a força e direção de uma relação linear entre as variáveis, que é o foco da correlação de Pearson. Desse modo, quanto maior a similaridade entre palavras, maior o valor da métrica de avaliação.

Uma relação monotônica é definida como: (i) Enquanto o valor de uma variável aumenta, o da outra variável também aumenta; ou (ii) enquanto o valor de uma variável diminui, o da segunda variável também diminui como mostrado na Figura 2 [Lund and Lund 2018].

Os experimentos foram realizados utilizando os vetores GloVe (*Global Vectors* pretreinados, de 300 dimensões treinados sobre o dump do Wikipedia (2014) + Gigaword 5 (6B tokens, 400K palavras). A abordagem GloVe constrói word embeddings de uma forma que a combinação dos vetores de palavras se relacione diretamente com a probabilidade dessas palavras ocorrerem na base. Seus *embeddings* podem ser interpretados como um resumo da base treinada com baixa dimensionalidade que reflete coocorrências [Pennington et al. 2014]. Em todos os experimentos realizados, aplicou-se o critério de redução para metade da dimensionalidade original, i.e., foram produzidos vetores com 150 dimensões.

Para avaliar e comparar os efeitos das diferentes estratégias de redução de dimensionalidade [Raunak 2017] e a truncagem, foram considerados os seguintes algoritmos:

Figura 2. Exemplo de relacionamento monotônico



1. **PCA:** Transforma os vetores de palavras utilizando o algoritmo PCA.
2. **P-PCA:** Transforma os vetores de palavras utilizando o algoritmo PCA após a aplicação do algoritmo de pós processamento.
3. **PCA-P:** Transforma os vetores de palavras utilizando o algoritmo PCA e então aplica o algoritmo de pós processamento.
4. **RAUNAK**: Transforma os vetores de palavras utilizando o algoritmo proposto por [Raunak 2017]

Os *embeddings* resultantes foram então truncados considerando 4, 6 e 8 bits, conforme descrito na Seção 2.2. Nos experimentos realizados, foram utilizadas as implementações do PCA disponível na ferramenta Scikit-Learn [Pedregosa et al. 2011]. As implementações dos algoritmos PPA e RAUNAK consideraram as versões disponibilizadas pelos próprios autores [Raunak 2017] .

4. Resultados

Os resultados individuais para cada combinação de método de redução de dimensionalidade e truncagem são apresentados na Tabela 1. Com exceção do Glove-300 mostrado na tabela, todos os outros arquivos foram reduzidos de 300, para 150 dimensões. Podemos observar que em algumas bases de dados, como por exemplo, o EN-MC-30, a aplicação do algoritmo PPA seguido do PCA, com truncagem de 4 bits apresentou desempenho melhor do que a representação original. O mesmo pode ser observado na base EN-VERB-143, na qual o algoritmo RAUNAK com truncagem de 6 bits também obteve o melhor desempenho.

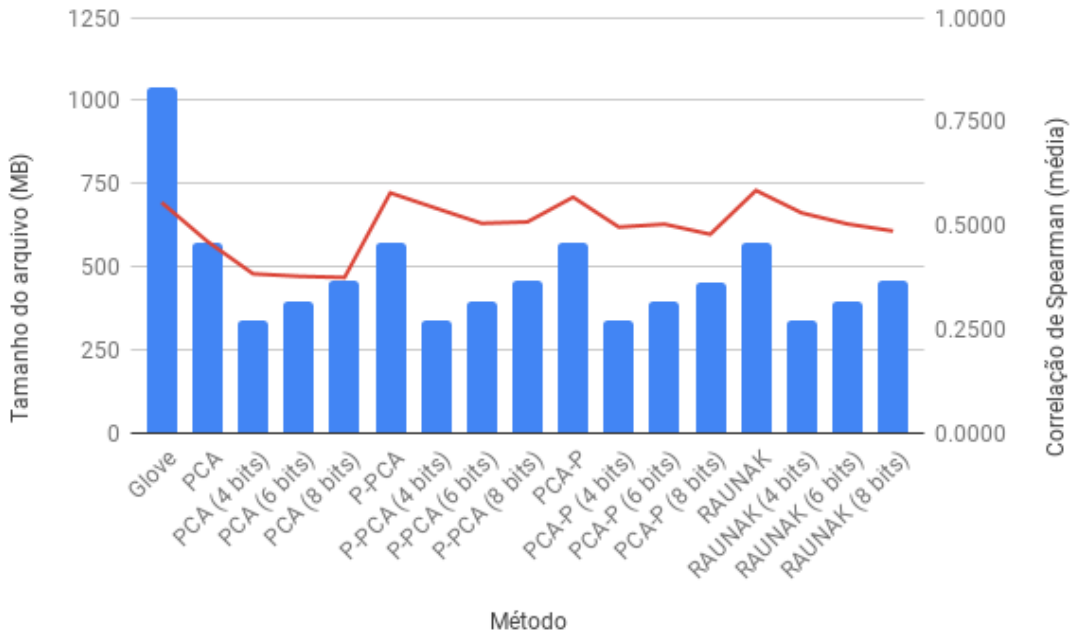
Através da tabela é possível perceber que dentre as três truncagens utilizadas (4, 6 e 8 bits), a truncagem de 4 bits obteve em geral melhores resultados, apresentando desempenho mais próximo (ou superior), ao original em 33 dos 52 experimentos. A truncagem de 8 bits, apresentada por [Ling et al. 2016] como suficiente para manter a performance em seu trabalho, quando combinada à estratégias de redução de dimensionalidade obteve a performance mais próxima da original em apenas 5 dos experimentos, sendo 4 desses obtidos através do algoritmo PCA. Nos outros 14 experimentos restantes, a truncagem de 6 bits foi a que mais se aproximou (ou obteve melhor desempenho). Esses resultados indicam que a diminuição no número de dimensões do vetor de representação vetorial, im-

pacta diretamente na redução do número de bits necessários para uma efetiva codificação de informação semântica.

Por outro lado, a redução do tamanho dos arquivos contendo as representações finais chega a ser bastante significativa, conforme apresentado na Figura 4, na qual se demonstra a correlação entre o tamanho do arquivo final e o desempenho médio (sobre todas as bases de dados analisadas) para cada combinação de método + truncagem.

Como esperado, os melhores resultados na redução de tamanho foram obtidos nos arquivos truncados com 4 bits. No melhor caso, onde o PCA foi utilizado após a aplicação do PPA, obtivemos uma redução de aproximadamente 40.73% no tamanho do arquivo, e uma perda média de apenas 6.39% na correlação de Spearman com as avaliações feitas por humanos. De maneira geral, o algoritmo proposto por [Raunak 2017] e a abordagem PPA-PCA obtiveram as maiores reduções de tamanho do arquivo (40,80% menores que o original). Apesar disso, é possível observar na Tabela 2 que as porcentagens de redução do tamanho do arquivo se mantiveram bastante semelhantes em todas as abordagens. A Tabela 2 também possui a diferença do desempenho médio (Correlação Spearman) dos métodos em cada base. A truncagem de 4 bits, em sua maioria, também resultou numa menor diferença em relação a média da Correlação Spearman dos arquivos sem truncagem.

Figura 3. Gráfico relacionando desempenho médio (linha vermelha) e o tamanho do arquivo com *embeddings*.



5. Conclusão

Os experimentos realizados complementam os estudos anteriores, ao combinar análises de redução de dimensionalidade com técnicas de redução de precisão dos vetores de *word embeddings* pré-treinados. Observou-se que a utilização da redução de dimensionalidade em conjunto com a redução da quantidade de bits necessária para representação de um

Tabela 1. Resultados individuais de cada arquivo de configuração

Dataset	*Embedding*	Sem truncagem	4 bits	6 bits	8 bits
EN-MC-30	Glove	0.7026	-	-	-
	PCA	0.6934	0.6554	0.6569	0.5690
	PCA-PPA	0.6917	0.6156	0.6289	0.5882
	PPA-PCA	0.7144	0.7255	0.6841	0.6899
	RAUNAK	0.7195	0.7177	0.5906	0.5788
EN-MEN-TR-3k	Glove	0.7375	-	-	-
	PCA	0.6251	0.5516	0.5505	0.5429
	PCA-PPA	0.7491	0.6911	0.6969	0.6767
	PPA-PCA	0.7539	0.7079	0.6847	0.6917
	RAUNAK	0.7531	0.7163	0.6943	0.6916
EN-MTurk-287	Glove	0.6332	-	-	-
	PCA	0.5429	0.5000	0.5443	0.4734
	PCA-PPA	0.6266	0.6053	0.6219	0.5949
	PPA-PCA	0.6594	0.6261	0.5384	0.6023
	RAUNAK	0.6199	0.5904	0.5700	0.5935
EN-MTurk-771	Glove	0.6501	-	-	-
	PCA	0.5263	0.4701	0.4571	0.4596
	PCA-PPA	0.6461	0.5843	0.5452	0.5745
	PPA-PCA	0.6491	0.5791	0.5905	0.5442
	RAUNAK	0.6483	0.6312	0.5572	0.5567
EN-RG-65	Glove	0.7662	-	-	-
	PCA	0.7283	0.6447	0.6896	0.6028
	PCA-PPA	0.7087	0.6956	0.6683	0.5066
	PPA-PCA	0.7501	0.7007	0.6647	0.6717
	RAUNAK	0.7784	0.6790	0.6034	0.6045
EN-RW-STANFORD	Glove	0.4118	-	-	-
	PCA	0.2839	0.2399	0.232	0.2244
	PCA-PPA	0.4166	0.3605	0.3428	0.3387
	PPA-PCA	0.4397	0.3701	0.3247	0.3375
	RAUNAK	0.4354	0.3712	0.3635	0.3529
EN-SIMLEX-999	Glove	0.3705	-	-	-
	PCA	0.2838	0.2142	0.1939	0.1959
	PCA-PPA	0.3591	0.2979	0.2932	0.2786
	PPA-PCA	0.3616	0.3215	0.3041	0.3173
	RAUNAK	0.3767	0.3405	0.3441	0.3158
EN-SimVerb-3500	Glove	0.2267	-	-	-
	PCA	0.1317	0.0974	0.0793	0.0776
	PCA-PPA	0.2525	0.1806	0.2065	0.1898
	PPA-PCA	0.2448	0.2065	0.2019	0.1772
	RAUNAK	0.2643	0.2310	0.2228	0.1829
EN-VERB-143	Glove	0.3051	-	-	-
	PCA	0.2883	0.1923	0.1636	0.2278
	PCA-PPA	0.3848	0.2013	0.3722	0.3043
	PPA-PCA	0.367	0.2748	0.3940	0.2161
	RAUNAK	0.4012	0.3496	0.4634	0.3213
EN-WS-353-ALL	Glove	0.6054	-	-	-
	PCA	0.4777	0.3531	0.3466	0.3917
	PCA-PPA	0.6652	0.6055	0.5722	0.5771
	PPA-PCA	0.6648	0.6513	0.5926	0.6136
	RAUNAK	0.6632	0.6125	0.5865	0.5911
EN-WS-353-REL	Glove	0.5725	-	-	-
	PCA	0.4272	0.3223	0.2982	0.3541
	PCA-PPA	0.6096	0.545	0.5093	0.5141
	PPA-PCA	0.6085	0.6005	0.5329	0.555
	RAUNAK	0.6085	0.5491	0.5101	0.5153
EN-WS-353-SIM	Glove	0.6638	-	-	-
	PCA	0.5469	0.4149	0.4253	0.4562
	PCA-PPA	0.7128	0.6545	0.6455	0.6450
	PPA-PCA	0.7250	0.7010	0.6432	0.6744
	RAUNAK	0.7189	0.6615	0.6397	0.6429
EN-YP-130	Glove	0.5613	-	-	-
	PCA	0.4333	0.3188	0.2607	0.2850
	PCA-PPA	0.5516	0.4004	0.4260	0.4245
	PPA-PCA	0.5693	0.5632	0.3997	0.5080
	RAUNAK	0.5968	0.4322	0.3899	0.3673

Tabela 2. Tamanho do arquivo e diferença do desempenho médio (Correlação Spearman) dos métodos em cada base

Embedding	Truncagem	Tamanho arquivo (MB)	Redução do arquivo %	Diferença Spearman
PCA	Sem truncagem	573,9	-	-
	4 bits	340,1	40,74	>16,93%
	6 bits	396,8	30,86	>18,21%
	8 bits	454,8	20,75	>18,84%
P-PCA	Sem truncagem	573,8	-	-
	4 bits	340,1	40,73	>6,39%
	6 bits	396,8	30,85	>12,68%
	8 bits	454,8	20,74	>12,10%
PCA-P	Sem truncagem	573,8	-	-
	4 bits	339,7	40,80	>12,70%
	6 bits	396,7	30,86	>11,46%
	8 bits	454,6	20,77	>15,75%
RAUNAK	Sem truncagem	573,8	-	-
	4 bits	339,7	40,80	>9,26%
	6 bits	396,6	30,88	>13,82%
	8 bits	454,8	20,74	>16,74%

vetor, de fato reduz significantemente o tamanho do arquivo resultante sem sacrificar muito a performance.

Esperamos que os resultados apresentados neste trabalho sirvam como referência para escolha de representações vetoriais de palavras para uso em dispositivos de memória reduzida e limitado poder computacional. Como trabalhos futuros, esperamos avaliar outras estratégias de redução de dimensionalidade, bem como avaliar os efeitos da redução de precisão durante o treinamento dos vetores, conforme indicado em [Ling et al. 2016].

Outro possível trabalho futuro é representar os vetores de *word embeddings* utilizando um sistema quântico [Nielsen and Chuang 2002]. Teoricamente a capacidade de armazenamento de informação em sistemas quânticos possui um ganho exponencial em relação aos computadores convencionais [Trugenberger 2001].

6. Agradecimentos

Este trabalho recebeu apoio do Instituto Serrapilheira (número do processo Serra-1709-22626).

Referências

Faruqui, M. and Dyer, C. (2014). Community evaluation and exchange of word vectors. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations.*, pages 19–24.

Goldberg, Y. (2017). *Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies)*. Morgan Claypool.

Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308.

Ling, S., Song, Y., and Roth, D. (2016). Word Embeddings with Limited Memory. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 387–392.

Lund, A. and Lund, M. (2018). Spearmans rank-order correlation.

Mu, J., Bhat, S., and Viswanath, P. (2017). All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Nielsen, M. A. and Chuang, I. (2002). Quantum computation and quantum information.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., and Others (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Raunak, V. (2017). Effective dimensionality reduction for word embeddings. *arXiv preprint arXiv:1708.03629*.

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Trugenberger, C. A. (2001). Probabilistic quantum memories. *Physical Review Letters*, 87(6):067901.

Well, A. D. and Myers, J. L. (2003). *Research design & statistical analysis*. Psychology Press.